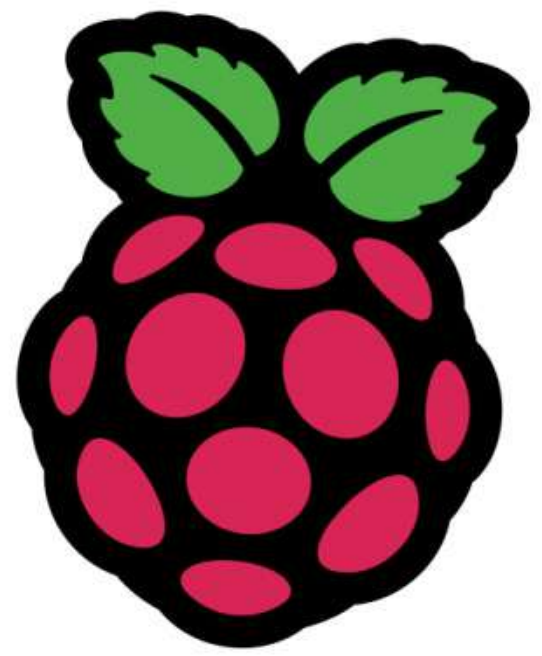




BUY IN PRINT **WORLDWIDE** [MAGPI.CC/STORE](https://magpi.cc/store)

The *MagPi*



Issue 119

July 2022

magpi.cc

The official Raspberry Pi magazine

NEW!

RASPBERRY PI

PICO W

+ RASPBERRY PI SILICON

+ WIRELESS LAN

+ \$6 AVAILABLE NOW!

SET UP A
**WEATHER
STATION**

CREATE KITT
FROM KNIGHT RIDER

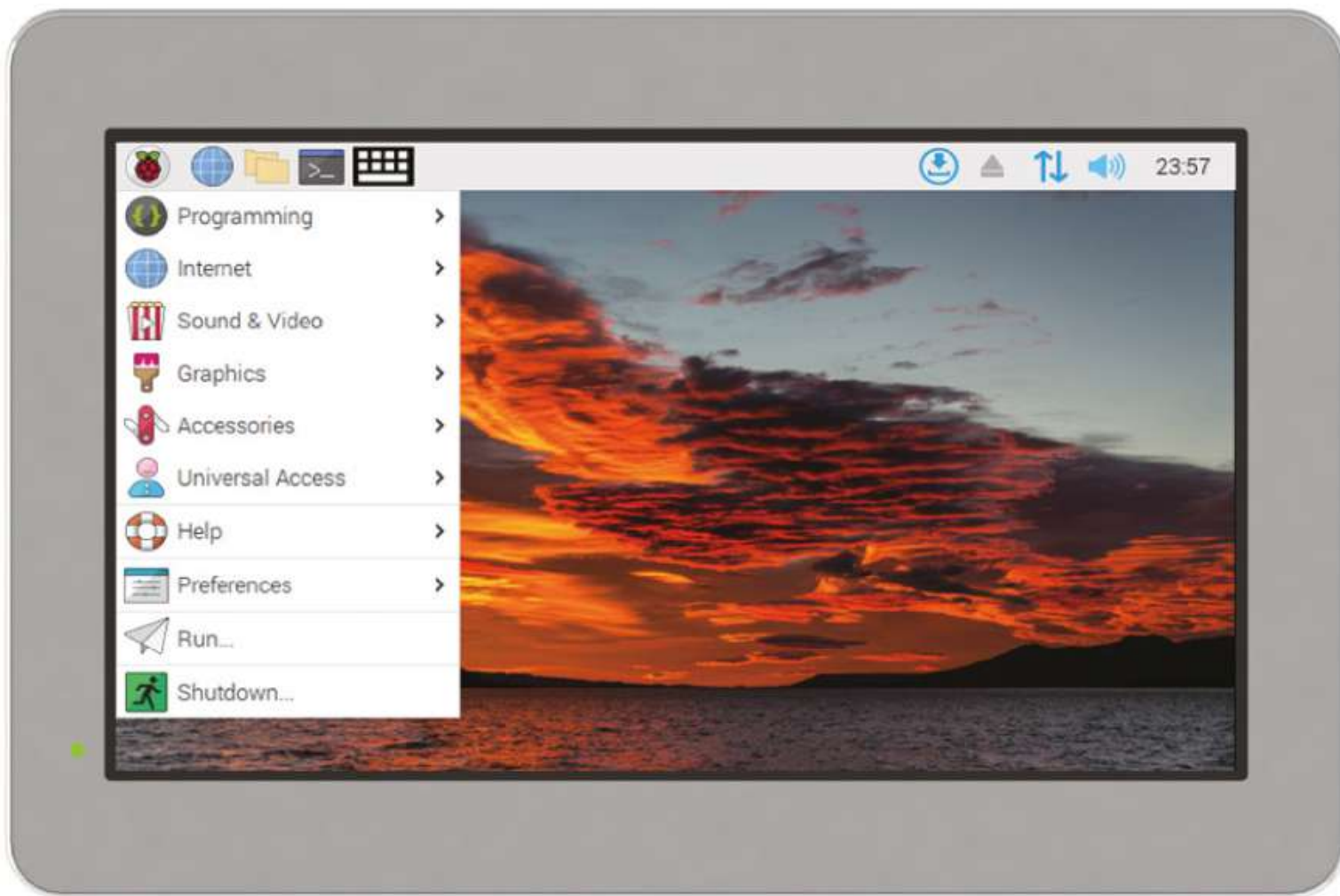


**GIANT
SPIDERPI
TESTED!**

GREAT PROJECTS FOR ORIGINAL RASPBERRY PI COMPUTERS



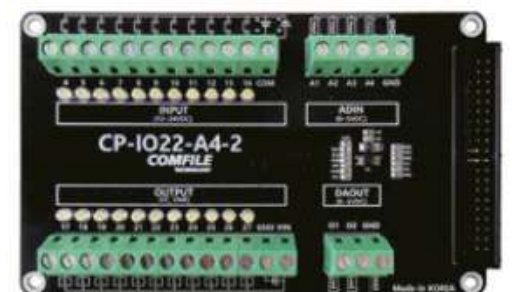
Industrial Raspberry Pi



ComfilePi

The ComfilePi is a touch panel PC designed with high-tolerant components and no moving parts for industrial applications. It features a water-resistant front panel, touchscreen, color LCD (available in various sizes), RS-232, RS-485, Ethernet, USB, I2C, SPI, digital IO, battery-backed RTC (real-time clock), and piezo buzzer.

Use the rear-panel 40-pin GPIO header to expand its features and capabilities with additional I/O boards. The ComfilePi is UL Listed and employs Raspberry Pi Compute Module.



WELCOME to The MagPi 119

I have been playing with Pico W for a few weeks now. One of the perks of the job is early access to technology.

Pico W is astounding. Pico is already a miniature marvel that packs tremendous power into a small space. Fitting both a wireless LAN chip and an antenna required some extremely clever engineering. You can read all about it on page 34.

Beyond the very clever design, putting wireless functionality on the Pico microcontroller is a game-changer. Now Pico can hook up to your network, make requests from online APIs, and send back data readings to a networked computer.

This magazine is packed with incredible projects built by Raspberry Pi's vibrant community. From recreations of KITT from *Knight Rider* (page 18) to building your own Weather Station (page 46). People have been building incredible things with Raspberry Pi and Raspberry Pi Pico for years, and now they'll make even better projects with a wireless Raspberry Pi Pico W.

I can't wait to get started.

Lucy Hattersley Editor



EDITOR Lucy Hattersley

Lucy Hattersley is editor of *The MagPi* magazine and just spent a week in Mablethorpe, and is now readjusting to London prices.

@LucyHattersley

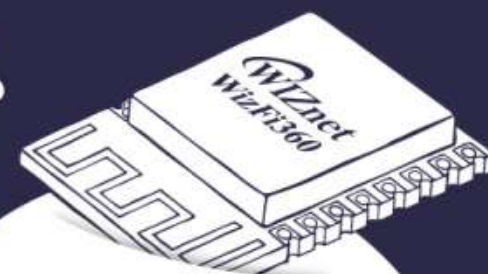
GET A
**RASPBERRY PI
PICO W**
WITH A SUBSCRIPTION!
PAGE 28



WizFi360

Design Contest

at maker.wiznet.io



WizFi360 with

ARMKEIL

Raspberry Pi RP2040

ARDUINO

Bring your **creativity** and win an **Apple iPad Pro**

WizFi360

- Official Wi-Fi Shield on ARM Open-CMSIS-Pack and Keil Studio Cloud
- Easy-to-connect Wi-Fi to Pico RP2040
- Support Azure, AWS SDK Example

Free Sample

More details at maker.wiznet.io



WizFi360

or



WizFi360-EVB-Mini

or



WizFi360-EVB-Pico



Contest begins

June 20



Apply & get a free sample

July 31



Submissions close

September 30



Winners announced by

October 17



30 Winners will receive **Apple iPad Pro(\$1,099)** 12.9-inch 128GB Wi-Fi

Contents

➤ Issue 119 ➤ July 2022

Cover Feature

34 Raspberry Pi Pico W

Regulars

30 Case Study: Korg

92 Your Letters

97 Next Month

98 The Final Word

Project Showcases

08 Little Sun Gazer

10 GroundBIRD telescope

14 Saxophone light

16 OpenMower

18 KITT

20 Macintosh SE/30 Raspberry Pi

22 BirdNET-Pi

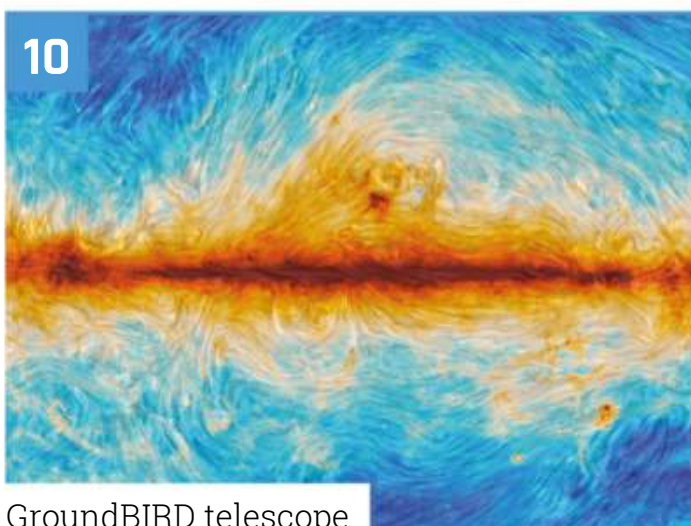
26 Night Camera



34



16



10

GroundBIRD telescope

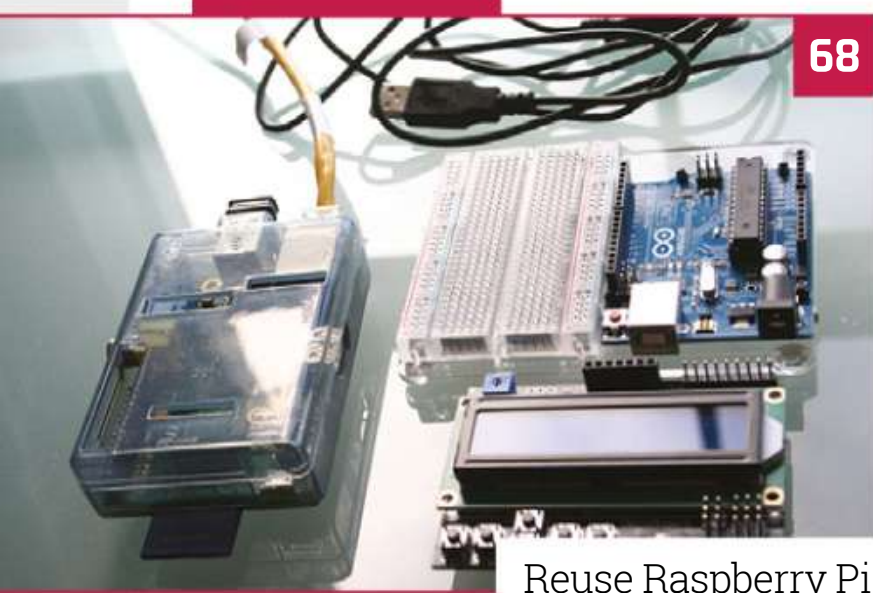
OpenMower

The MagPi magazine is published monthly by Raspberry Pi Ltd. Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701, is the mailing agent for copies distributed in the US. Application to mail at Periodicals prices is pending at Williamsport, PA. POSTMASTER: Send address changes to The MagPi magazine c/o Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701.

Tutorials

- 46** Build a weather station
- 50** Learn ARM Assembly - part 4
- 54** Digital Analogue Converter
- 58** ArtEvolver - part 2
- 64** Code an homage to Game & Watch

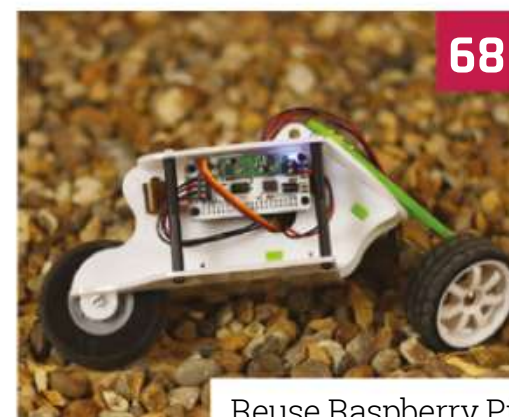
The Big Feature



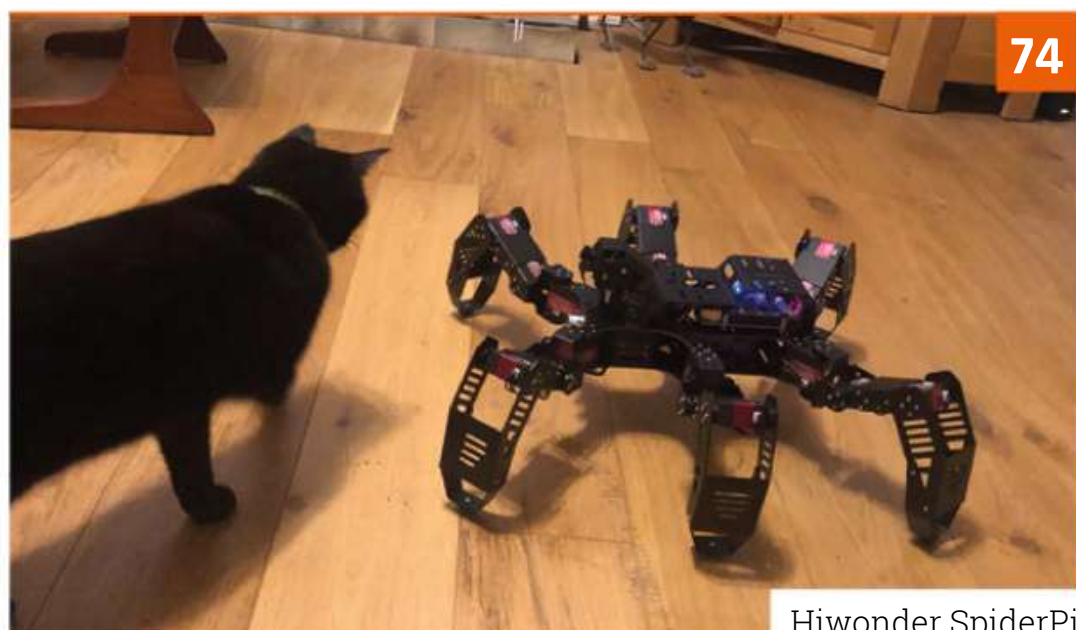
Reuse Raspberry Pi



Build a weather station



Reuse Raspberry Pi



Hiwonder SpiderPi

Reviews

- 74** Hiwonder SpiderPi
- 76** Maker Nano RP2040
- 78** Pico Unicorn Pack
- 80** 10 amazing Summer projects
- 82** Learn databasing

Community

- 84** Super Make Something interview
- 86** Electromagnetic Field 2022
- 88** This Month in Raspberry Pi



EMF camp report

WIN
1 OF 20

RASPBERRY PI
PICO W



95

DISCLAIMER: Some of the tools and techniques shown in The MagPi magazine are dangerous unless used with skill, experience, and appropriate personal protection equipment. While we attempt to guide the reader, ultimately you are responsible for your own safety and understanding the limits of yourself and your equipment. Children should be supervised. Raspberry Pi Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in The MagPi magazine. Laws and regulations covering many of the topics in The MagPi magazine are different between countries, and are always subject to change. You are responsible for understanding the requirements in your jurisdiction and ensuring that you comply with them. Some manufacturers place limits on the use of their hardware which some projects or suggestions in The MagPi magazine may go beyond. It is your responsibility to understand the manufacturer's limits.

enviro goes wireless

Flexible Power
Pico W Aboard
Long Battery Life
Human Friendly
Real Time Clock

Alkaline NiMH Micro-USB LiPo

Wireless, standalone boards

Run for weeks or months on batteries

Working examples and custom libraries

Efficient sleep and scheduled readings



<https://pimoroni.com/envirow>

Automation 2040 ^W

RELAY AND IO CONTROL UP TO 40V WIRELESSLY
FOR HOME, OFFICE AND INDUSTRIAL AUTOMATION

PICO W
ABOARD



<https://pimoroni.com/auto2040w>

INKY FRAME 5.7

Your seven colour E Ink pal who's fun to be with



<https://pimoroni.com/inkyframe>

PICO W
ABOARD

PIMORONI

Little Sun Gazer

Want to keep an eye on the sun without blinding yourself?
Then fix your peepers on this hot device, as **David Crookes** explains



MAKER Dmytro Panin

Dmytro Panin is a programmer based in Ukraine, and he wrote his first line of code aged eight. He works for a large provider of nearshore software engineering services.

magpi.cc/sungazer

Ever since Dmytro Panin created his **Solar System Display to simulate the motion of the planets around the sun (see issue 110)**, he's had a burning desire to create a similar project to track and visualise our nearest star.

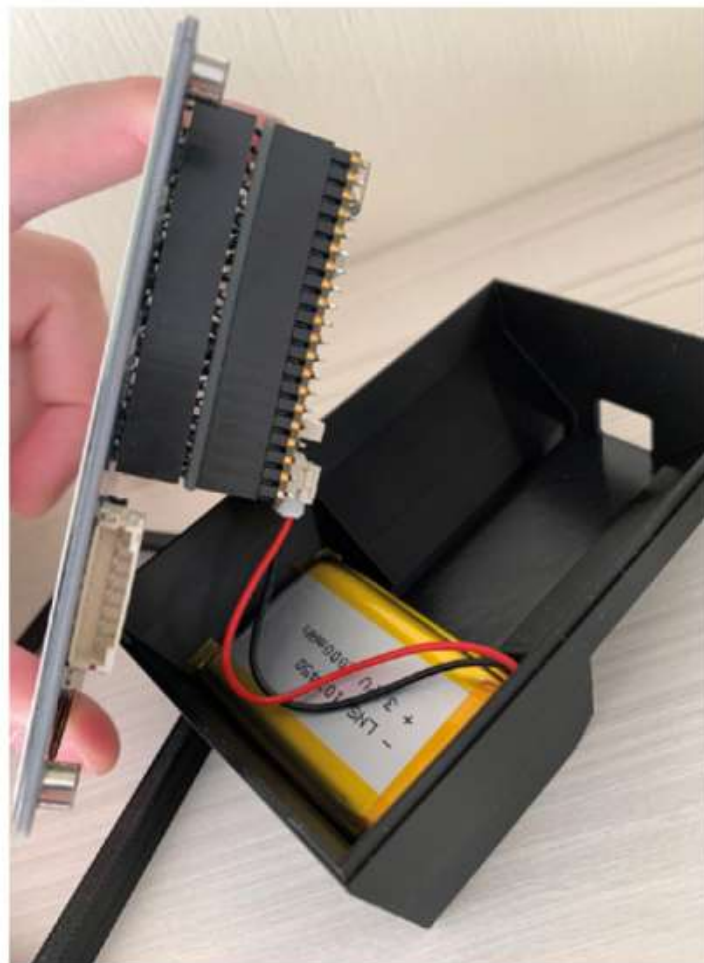
"One thing I knew I wanted to implement differently is how it was powered," he says, wanting his latest device to run off a battery. "Additionally, I realised I didn't get to explore the moon position in my previous project and I really wanted to visualise it too, and be able to see how it circles around the Earth as time passes by."

With that in mind, he got to work on a device that could calculate the sun's position based on a person's longitude and latitude. Aside from

showing the angle of the sun relative to where the viewer is on Earth, he wanted it to show the position of the Earth in relation to the star, and the position of the moon in relation to Earth. He also wanted to display the current time and show when the sun would rise and set.

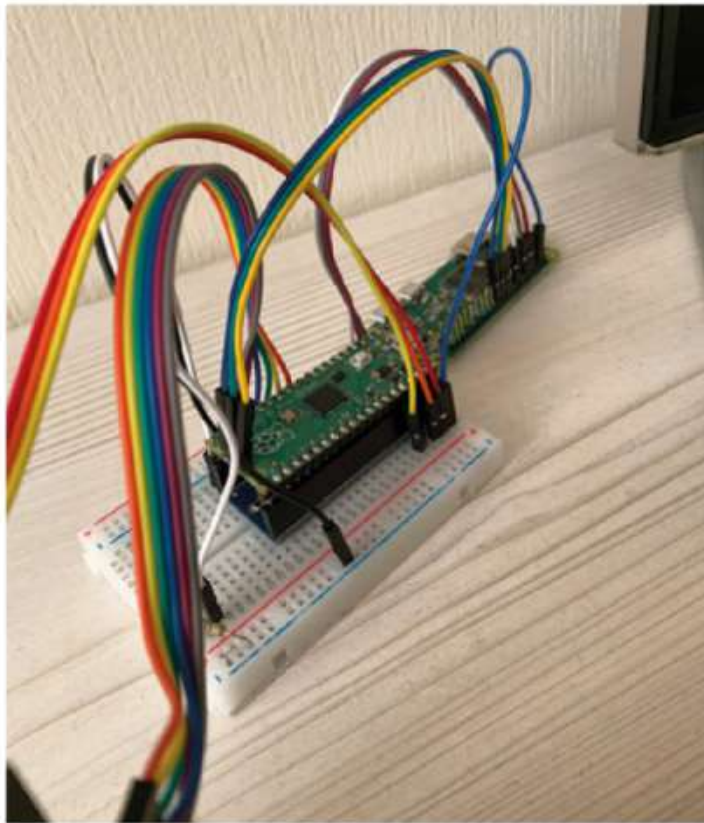
Powering on

As with the Solar System Display, he decided to use a Raspberry Pi Pico microcontroller board, not so much out of a desire to experiment, but out of practicality. "As the project was supposed to be battery-powered, I had to go with a microcontroller and not a system-on-a-chip computer such a Raspberry Pi Zero due to the power consumption limitations," he says.



► Dmytro says the code is messy, but the assembly of the components is most definitely not

▲ Dmytro has lots of plans for this build, and he's even considering a solar-powered device



▲ The project's progress has stalled due to the war in Ukraine. If you like the work so far, why not visit Dmytro's GitHub and buy him a coffee?

“I like the state this project is in, but it wasn't supposed to be the final iteration”

“Raspberry Pi Pico was my first choice because it has enough RAM to keep a frame buffer and a deep sleep mode which allowed me to save some battery. Additionally, I wanted to experiment with a 3.7-inch e-ink display and Waveshare had a ‘plug-and-play’ version of the display for Raspberry Pi Pico which cemented the choice.” A precision real-time-clock module completed the setup, along with a Pimoroni LiPo SHIM and Li-Po 2000 mAh 103450 rechargeable battery.

Holding on

The main work has involved calculating positions. “My background is in software engineering, and I always dabbled in astronomy, but astronomical formulas and calculations are definitely not my strong suit,” he says. “To become familiar with the subject and put together the model, I had to consult multiple sources and examples such as *Astronomical Algorithms* by Jean Meeus, *Computing Planetary Positions* by Paul Schlyter (magpi.cc/planetpositions), and the National Oceanic and Atmospheric Administration website (noaa.gov).”



The project uses the Waveshare 3.7-inch e-Paper e-Ink Display HAT which has a resolution of 480x280

The astronomical calculations are based on the viewer's location and time

The display sits in a 3D-printed case and the battery, according to Dmytro, should last four weeks between charges

Although he has plans to create a 3D-printable case, which he aims to put on MakerBot Thingiverse, the project is currently on hold. “There are a number of things I was looking to improve, starting from the case and ending at the diagrams/widgets/functionality, but I had to stop working on this project quite abruptly,” he says. As regular readers will know, Dmytro lives in Ukraine and he had to flee Kyiv with only a backpack of possessions when the shelling began.

“I like the state this project is in now, though,” he continues. “But it wasn't supposed to be the final iteration. For example, I wanted to show and incorporate noon – the code for it has been written but I never got to incorporate it in the diagram. Another example could be that there is still no easy way to set time, so this is the first thing I need to address once I get to work on it. I hope I will be able to continue working on it soon.”

Quick FACTS

- ▶ The device is battery-powered
- ▶ It displays the sun's position in real-time
- ▶ It's based around Raspberry Pi Pico
- ▶ The battery powers the microcontroller board and display
- ▶ The code is incomplete at the moment

GroundBIRD telescope

An international group of astronomers developed a telescope that investigates the very early universe. **Rosie Hattersley** learns about its inner workings



GroundBIRD is a Raspberry Pi-enabled millimetre-wave telescope that observes polarisation patterns of the cosmic microwave background, which in turn reveals information about the very early universe and its large-scale properties.

GroundBIRD's telescope has an amazingly high rotation speed of 120° per second, and is kept cool to near absolute zero temperature.

"The telescope is designed to achieve the highest sensitivity for wide sky observation, with a continuous high rotation speed of 120°/s to suppress atmospheric fluctuations," explains Mike Peel, part of a team of astronomers from

Japan, Korea, Spain, and The Netherlands who developed and work on the GroundBIRD project.

Achieving precise observations with the telescope involves multiple Raspberry Pi monitors keeping an eye on the telescope's hardware operations, such as controlling its rotation speed and the temperature, humidity, and pressure in the telescope dome, in addition to monitoring the prevailing weather conditions.

Communication hub

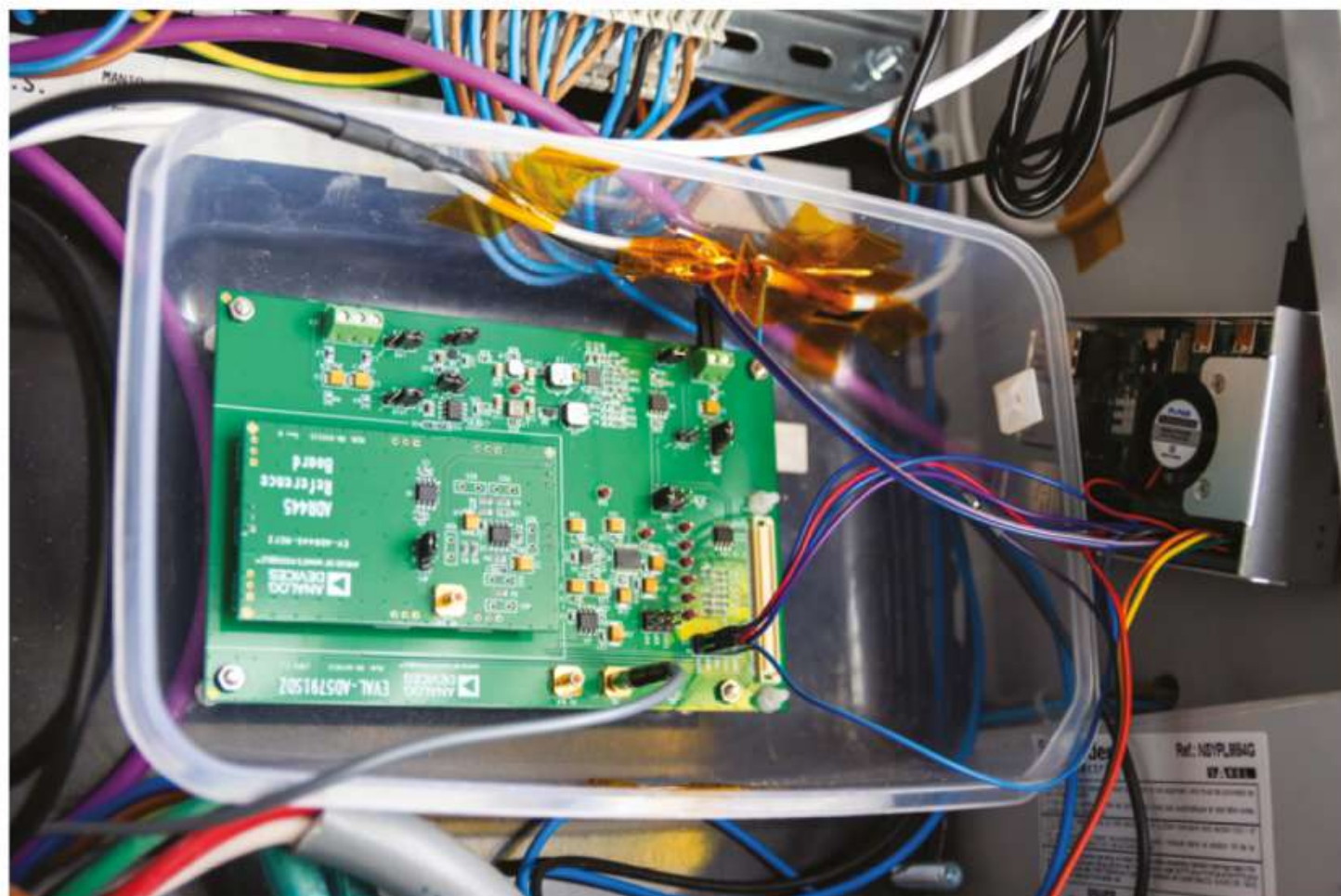
GroundBIRD uses a number of Raspberry Pi computers – one for each hardware/sub-project evaluation – each of which is developed and

MAKER

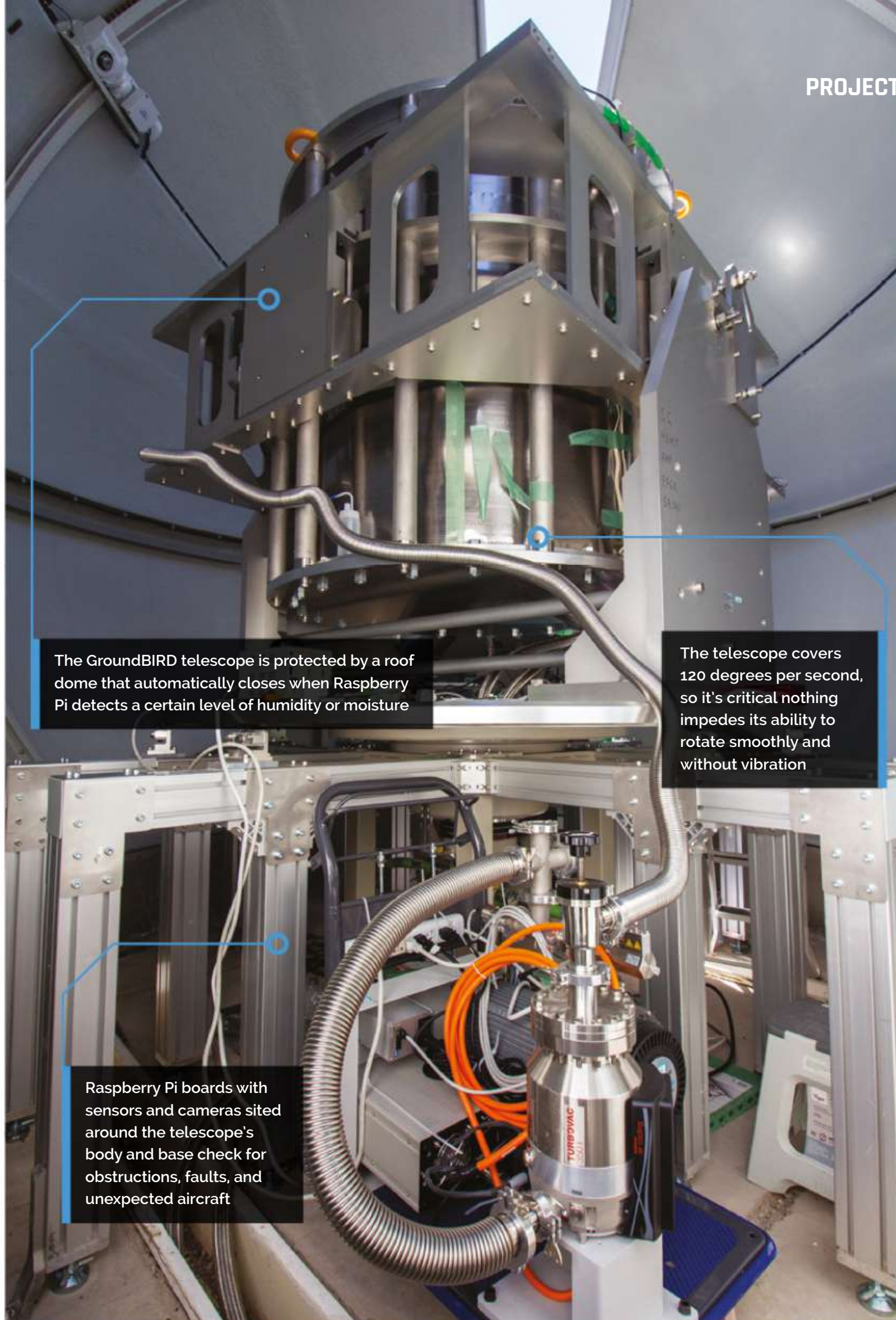
GroundBIRD

GroundBIRD consists of developers from Japan, Korea, The Netherlands, and Spain who each developed telescope sensor modules based on Raspberry Pi.

magpi.cc/groundbird



▶ The Raspberry Pi, yuzu-pi controlling the azimuth rotation through a DAC board



Quick FACTS

- ▶ Accurate weather information is provided by a Bosch BME280 sensor
- ▶ It also records volcanic vibrations from around the world
- ▶ Another Raspberry Pi sensor detects changes to vibrations on the telescope's leg
- ▶ Calima dust from the Sahara often coats the telescope...
- ▶ A good argument for having a dome that can be closed against the elements!

Image credit: Mike Peel

tested independently before all the hardware is installed directly into the telescope. This approach meant troubleshooting one Raspberry Pi-enabled feature didn't hold back progress with other elements of the project. The sensors, monitoring instruments, and FPGA (field-programmable gate array) circuitry were also bought as off-the-shelf components and adapted for the project's needs using the wealth of useful information readily available online about how to use Raspberry Pi for specific purposes. Much of the development work

involved ways to enable modules to communicate with the Raspberry Pi server.

The GroundBIRD telescope was initially developed and constructed in Japan and transported to the Teide Observatory in the Canary Islands in 2018. Code written for the project was first shared locally using GitLab, but will eventually be available on GitHub. Communication is over a wired Ethernet, since wireless connections would interfere with the radio observatory's signals. Python scripts from sensors or cameras connected to each Raspberry Pi relay



▲ Shunsuke Honda points out the pera-pi Raspberry Pi sensor checking on the humidity and temperature within the telescope dome

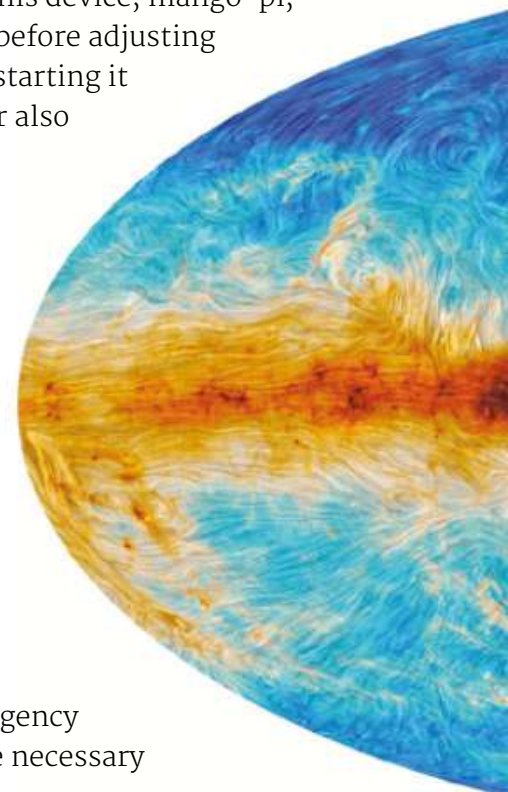
information in real-time to the GroundBIRD server which consists of a desktop PC, two rack-mounted PCs, and seven Raspberry Pi computers. The dome is controlled solely by Raspberry Pi, whereas the humidity detector – which lets the researchers know it's too damp to have the dome open – runs a different system.

The team assigned each of the six discrete Raspberry Pi boards a name, either of a fruit or a ground bird, so mango-pi, yuzu-pi, pera-pi weather sensor, kaki-pi, choco-pi, and momo-pi. Most of these were adapted and up and running in the space of a few days, whereas the more complex FPGA coding for the angle encoder took several months of development. This was also the most costly element, aside from the hardware associated with the telescope itself.

Fruits of their labour

Although space is limited in the dome, the team was keen to install a very small PC and display.


Raspberry Pi and a small touchscreen display seemed an obvious choice. This device, mango-pi, checks all DAQ connections before adjusting the telescope's elevation or starting it moving. This same computer also controls the dome while the main server for GroundBIRD observation continuously rotates together with the telescope, and therefore cannot be connected from the dome control. Another Raspberry Pi, yuzu-pi, is fitted with a DAC HAT and has an SPI connection to the telescope's azimuth controller. By changing the voltage to the azimuth controller, it acts as an emergency stop mechanism should it be necessary



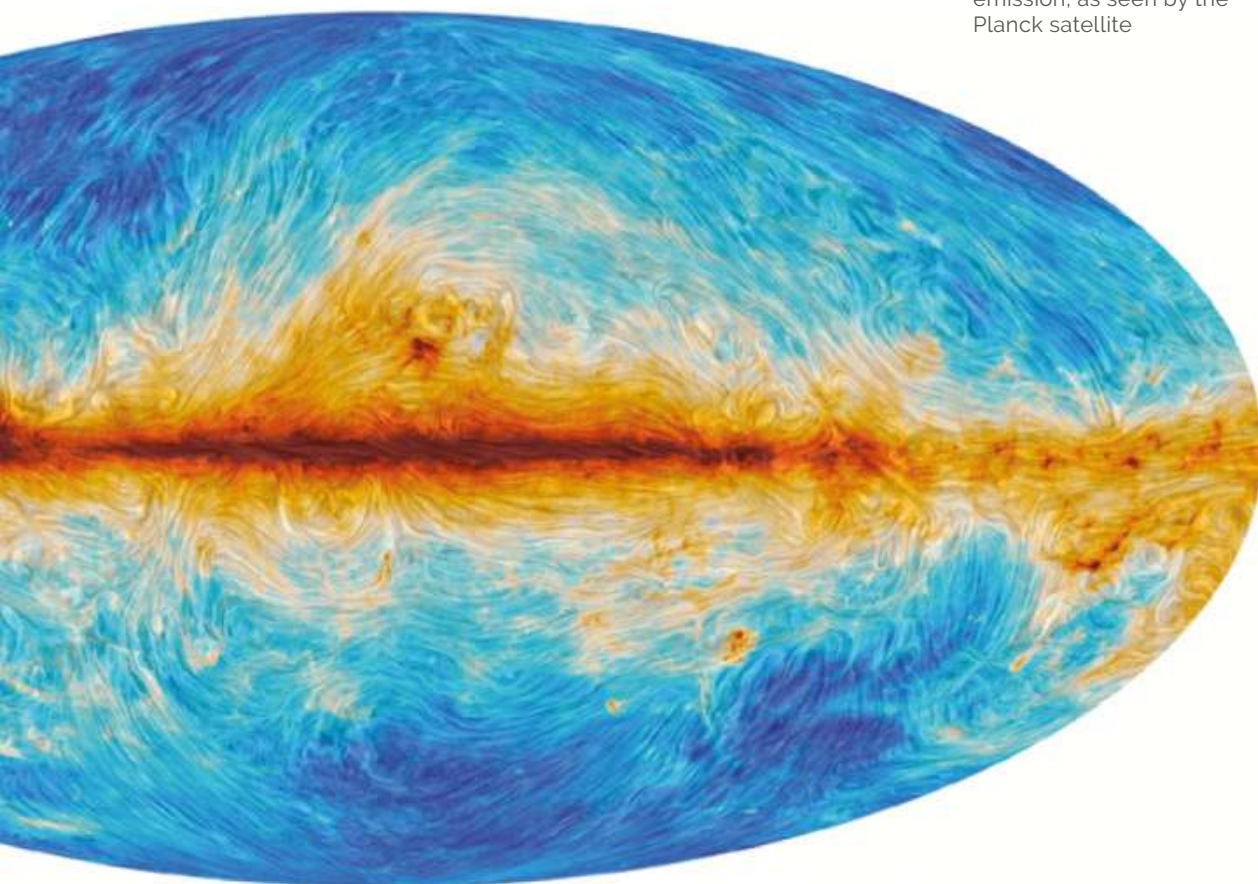
to prevent the fully remotely-controlled telescope from continuing to rotate.

Choco-pi trains four cameras on the telescope's wheels, while momo-pi listens for unexpected noises related to the telescope's rotation and acts as an early warning system for potential problems. "The pulse-tube-cooler (the cryostat cooling down from the room temperature to 4 Kelvin [-269°C] inside the telescope) also generates periodical sound," Mike explains. "It would be nice to

“ The telescope has an amazingly high rotation speed of 120° per second and is kept to close to absolute zero temperature ”

measure this to see whether there is something odd around the cryocoolers.” Finally, coco-pi is fitted with an ADS-B receiver and keeps an eye out for aeroplanes. The observatory is in a no-fly zone, so few aircraft are likely to pass by. However, its high altitude means the small antenna is able to detect aircraft up to around 500 km away. 

▼ The whole of the sky in thermal dust emission, as seen by the Planck satellite



Scanning the skies



01 GroundBIRD makes use of multiple cameras, sensors, and weather stations, each of which ensures the continuous effective running of the host telescope. Data from each is sent to a host Raspberry Pi server.



02 A humidity monitor was set up for the telescope because the DAQ [data acquisition] system originally had no roof. A Raspberry Pi initiates the dome closure whenever humidity gets too high.



03 Additional Raspberry Pi sensors and cameras trained on the telescope's wheels and exterior provide an early warning of faults or vibrations beyond the delicate tolerance levels.

Saxophone Light

Music is magical, especially with automated lights accompanying it.

Rob Zwetsloot conducts an inspection



Keith Thomson

A retired HR director and former engineer who likes interesting projects and supporting education.

What are tech friends for if not helping you make cool, reactive lights for your musical instruments? In the case of Keith Thomson and his friend Chris Croft, it was updating a previous version.

“Chris Croft, a friend of over 40 years is not only the world’s most watched project management trainer, but plays saxophone in a band (lost-at-sea.co.uk) for fun,” Keith explains.



▶ Chris playing his newly Raspberry Pi-powered saxophone

“He had someone design a light for his saxophone to illuminate it based on the note he was playing. It’s a great piece of electronics, but crucially chews through batteries (he has to replace them in the interval), is bulky, meaning that he has to physically wear the electronics, and is expensive to modify as it is based on complex circuitry with quite a few PCBs. I thought I could come up with a new software-based design that overcame all the problems that were bugging Chris. I also wanted it to be low-cost.”

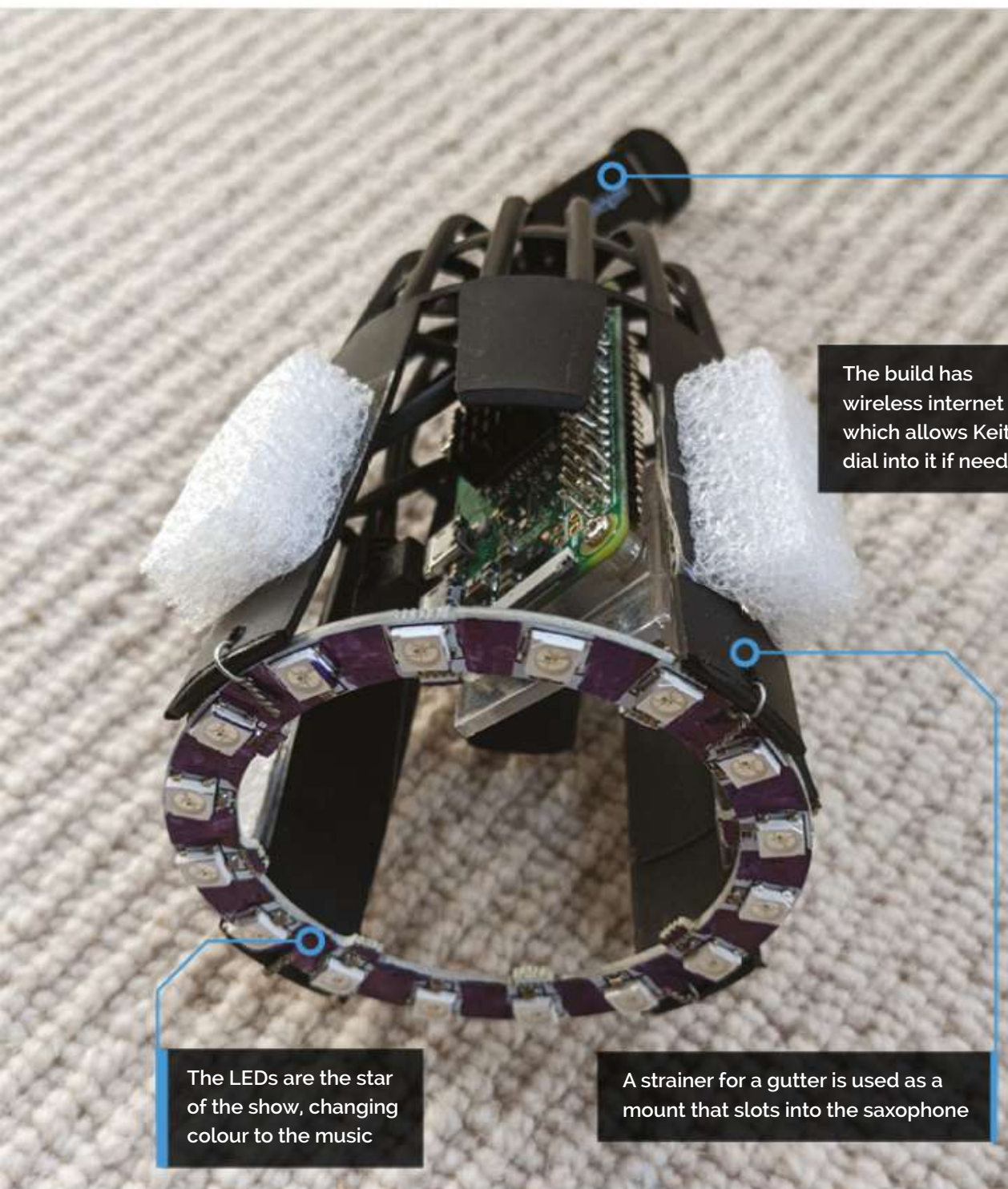
Smaller solution

Keith decided to go with Raspberry Pi, not only because he’d used it before in projects, but also because Raspberry Pi Zero was small enough to fit in the bell of the saxophone.

“The design process turned out to be quite complex and involved a lot of dead ends,” Keith admits. “For example, one design ended up with too much interference from the LEDs on the microphone, so I had to abandon it. The main issues involved finding a suitable microphone that was not susceptible to interference and would work with a Raspberry Pi, finding LEDs that could be easily controlled and interfaced, setting up the code to reliably detect saxophone notes and ignore the rest of the band, and, finally, fine-tuning the lighting to remove flicker and fade appropriately. The physical build was relatively simple, and involved modifying a strainer from a gutter to house the components, including a switch to safely turn Raspberry Pi off at the end of a gig.”

Jam session

So, how does Chris like the final product? “Chris loves it!” Keith tells us. “It’s a game-changer



Quick FACTS

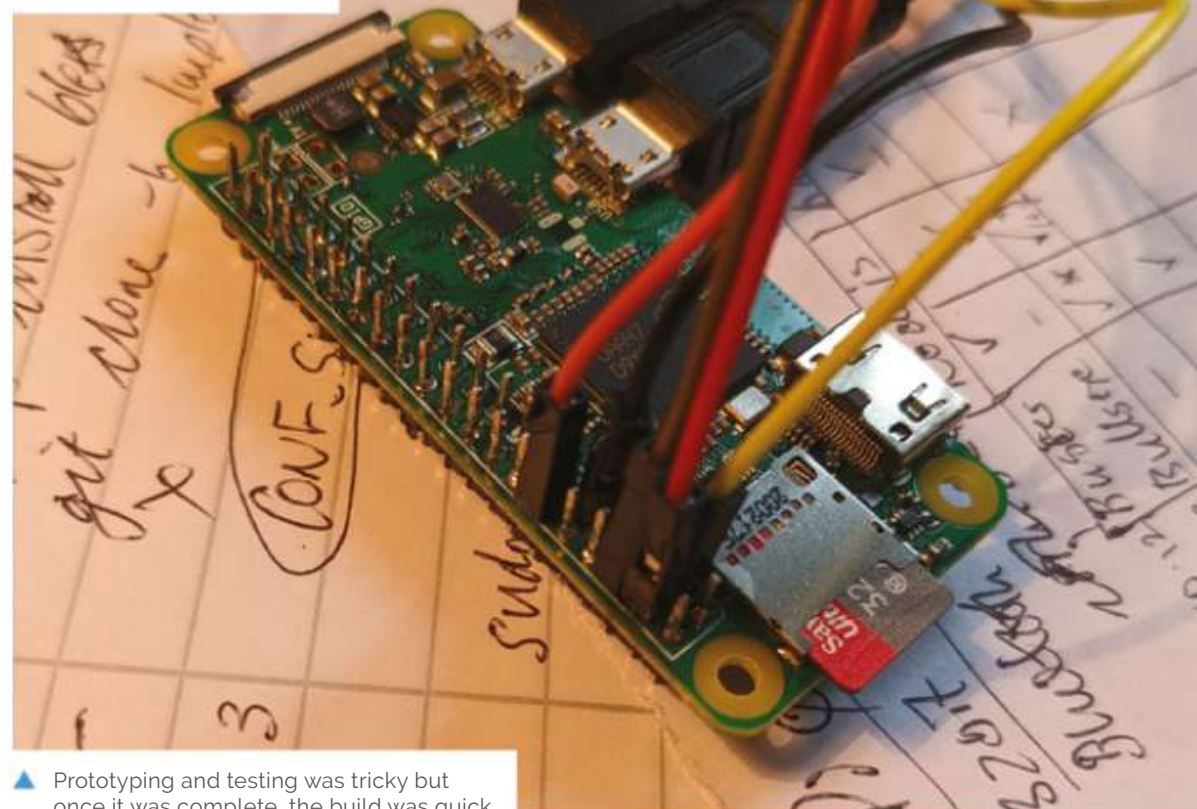
- ▶ It uses a mini USB microphone to listen to the sax
- ▶ The battery pack is mounted outside of the saxophone
- ▶ The LEDs used were NeoPixels
- ▶ The part the light is contained in is called the bell
- ▶ The pitch sets the colour, while volume sets the brightness

for him and not only overcomes the previous problems, but performs better too (more colours, less flicker, less interference from other sounds and so on).”

“ It’s a game-changer for him and not only overcomes the previous problems, but performs better too ”

While there are currently no upgrade plans for the light, Chris is going to test it further at future gigs. Keith can always remotely dial into it anyway thanks to VNC.

We hope to one day see more light-up instruments at the gigs we go to. **M**



OpenMower

If the thought of keeping your lawn magnificently manicured leaves you feeling unmotivated, help is at hand. **Nicola King** meets her new best friend



Clemens Elflein

Clemens is a self-employed software engineer who studied computer science and also likes to tinker with electronics and robots.

x-tech.online



Alert! Lithium & Blades

This project uses the mower's stock lithium battery, 29.4V when fully charged. Be very careful with such batteries, since they are dangerous if overcharged or shorted. Also, be careful when working on machinery with spinning blades! Always make sure the power is disconnected while you work.

magpi.cc/batterysafety

There's a lot to be said, from a helping wildlife perspective, for leaving parts of your garden to grow completely wild.

However, many people still yearn for that well-groomed, trim lawn which, unfortunately, takes both time and effort. Enter OpenMower, a nifty little gadget that will eliminate some of the strain and pain involved in cultivating the perfect lawn.

Surprisingly, creator Clemens Elflein's initial motivation for the project was not lawn care at all. "I noticed that the options for high-quality robotics prototyping solutions are slim," he tells us. "There are many small robotics kits available which feature an Arduino-based controller, two motors, and maybe some ultrasonic or collision sensors. These kits are affordable and great for getting something to move without much experience, but won't get you far in terms of playing with robotics algorithms on a higher level."

So, wanting to take things up a notch, Clemens began to work on OpenMower's architecture, which he sees as, "basically the first tiny step in the right direction: it allows me to evaluate off-the-shelf hardware performance (RTK GPS, motor

controllers, etc.) while also allowing me to play with the software side of things. As a bonus, it performs a useful task."

Even so, his end goal is to provide a robotics hardware prototyping platform which can bridge the gap between low-cost indoor prototyping to a proof-of-concept product. "I'm not looking to build the best lawn-mowing robot out there, but to build a thought-out robotics platform which can also be used as a lawn-mower," he emphasises.

Marvellous mowing machine

For OpenMower, Clemens took an off-the-shelf robotic mower and replaced the electronics with his own custom hardware blend. It uses a Raspberry Pi 4 as its main processor, and a Raspberry Pi Pico as a low-level controller.

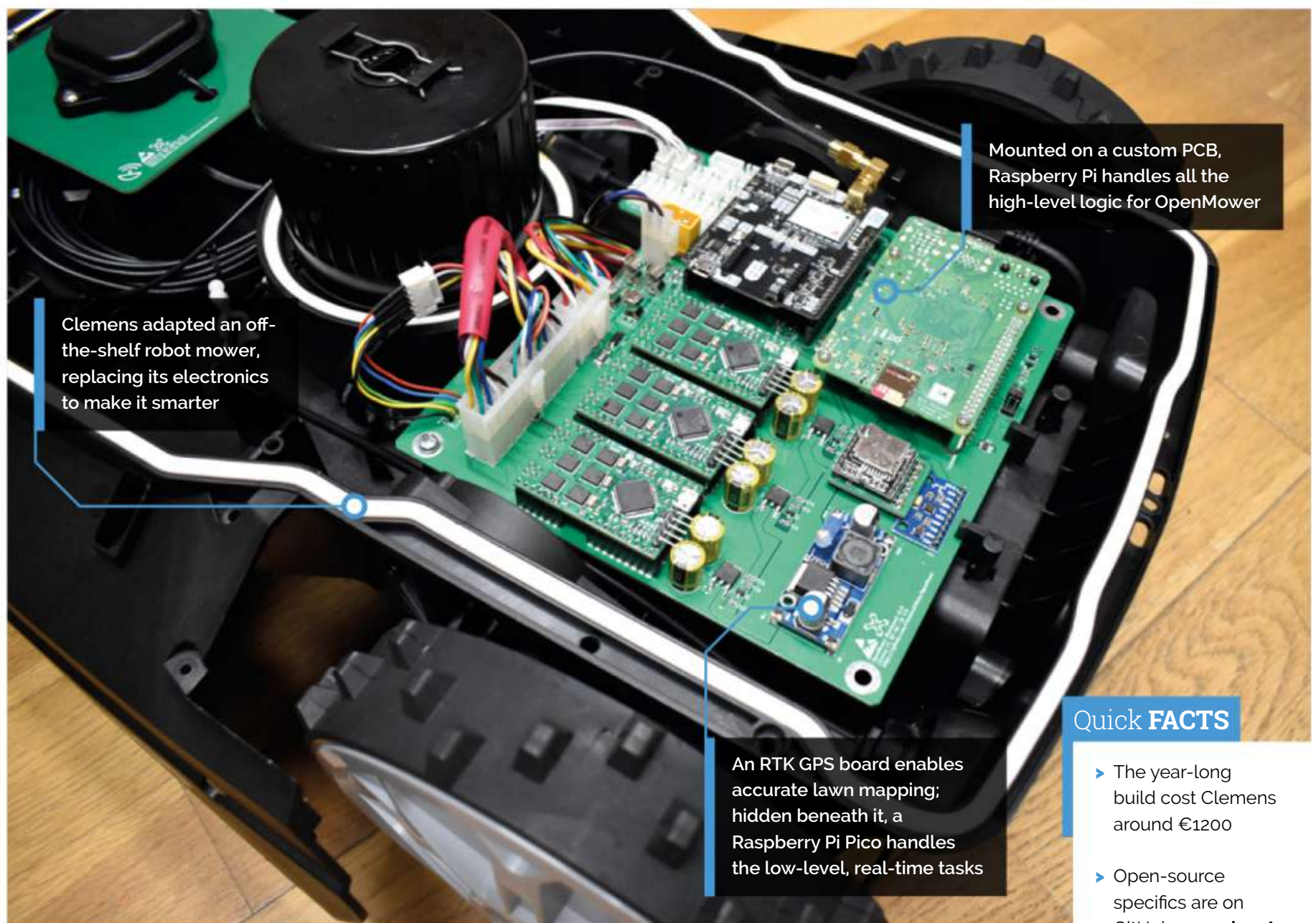
"The low-level, real-time, critical tasks, like checking the emergency stop switches, reading IMU and ultrasonic sensors, checking the battery voltages etc. is done by Pico in parallel, and the results are sent to the larger Raspberry Pi via UART," explains Clemens. This enables Raspberry Pi 4 to focus on high-level logic like positioning, navigation, coverage path calculation, and so on.



▲ By first driving the robot around the lawn perimeter, and obstacles such as trees, a GPS map is built up for automatic mowing



▲ Once set up with a GPS map, the OpenMower runs automatically, skirting the perimeter before mowing in a linear pattern



Quick FACTS

- ▶ The year-long build cost Clemens around €1200
- ▶ Open-source specifics are on GitHub: magpi.cc/openmowerrgh
- ▶ The mower takes around an hour to mow his lawn
- ▶ See it in action on YouTube: magpi.cc/openmoweryt
- ▶ He has also made a Raspberry Pi-based photo booth: self-o-mat.de

Clemens also added precise RTK (real-time kinematic positioning) GPS to his bot which, he says, makes it very flexible, efficient, and allows for better control. It even means that you can have multiple areas mowed separately. “You can plan an efficient path which covers the area, and even pause and resume exactly where you left off.”

“Once the setup is complete, it will mow away happily”

With the GPS’s u-blox chip outputting the robot’s position in XY co-ordinates relative to the base station with 1–2 cm precision, Clemens can first build an accurate map by driving the robot manually around the areas which need to be mowed. “When it’s time to mow, the OpenMower software internally uses the Slic3r library, which is basically a tool path planner for 3D printers, to plan a coverage plan for the recorded area.”

Significantly, it is worth noting that there is no need for any perimeter wires to inform the bot where to stop and start mowing, as this little bot is much smarter than that – once the setup



▲ A Yard Force Classic 500 mower was used as a base for the project, providing the required motors and mowing components in a neat package

is complete, it will mow away happily in a linear pattern using the coverage plan.

While his OpenMower works well for him, Clemens believes that it’s too early to say if it’s a robust solution for long-term use: “We’ll see that as soon as some members of the community have more experience with their own version of the mower.”

That sounds like an invitation to make your own version... go on, give it a mow! 🌱

KITT

Knight Rider is now 40 years old and Fred Arias has created the perfect birthday present, as **David Crookes** discovers



Fred Arias

Fred Arias is currently living abroad and likes to spend his free time travelling, brewing beer, and learning how to code to build fun and useful projects.

magpi.cc/kitt

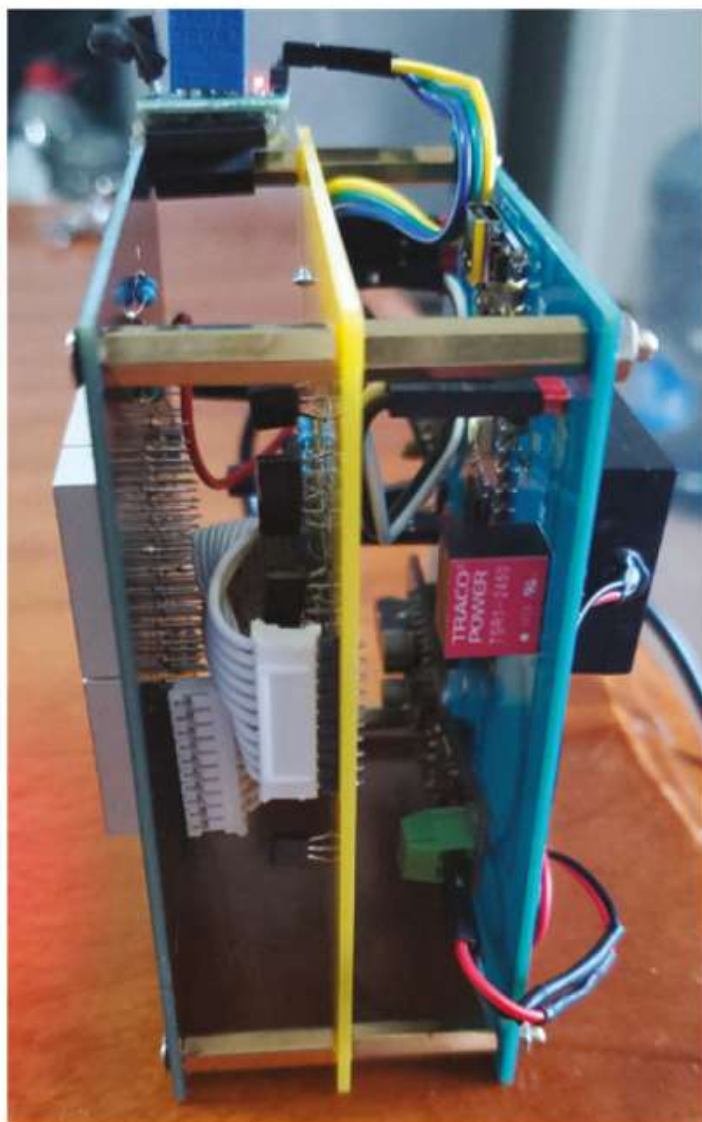
Although David Hasselhoff became a household name playing Michael Knight in the hit 1980s TV show *Knight Rider*, the main attraction was undoubtedly KITT, his cool car co-star. Quite aside from being a sleek black 1982 Pontiac Firebird Trans-Am, the vehicle drove straight to viewers' hearts because it was a talking supercomputer on wheels with a dry sense of humour.

"As a kid during the 1980s, I was a big fan of *Knight Rider* and I remember watching it thinking KITT was the coolest car ever," says Fred Arias. "I even remember going to Universal Studios in Hollywood, where I got the opportunity to sit inside KITT and talk to him directly. That experience never left me."

Decades later, Fred has recreated his own version, making use of an RP2040 microcontroller. "Over the years, I've seen people recreate KITT in the form of a VU meter but always needing to provide an audio source manually," he says. "I've been playing with a Raspberry Pi Pico over the last few months, while learning MicroPython to control different hardware, and I thought it would be best to learn how to use it by incorporating it into fun projects."

On a crusade

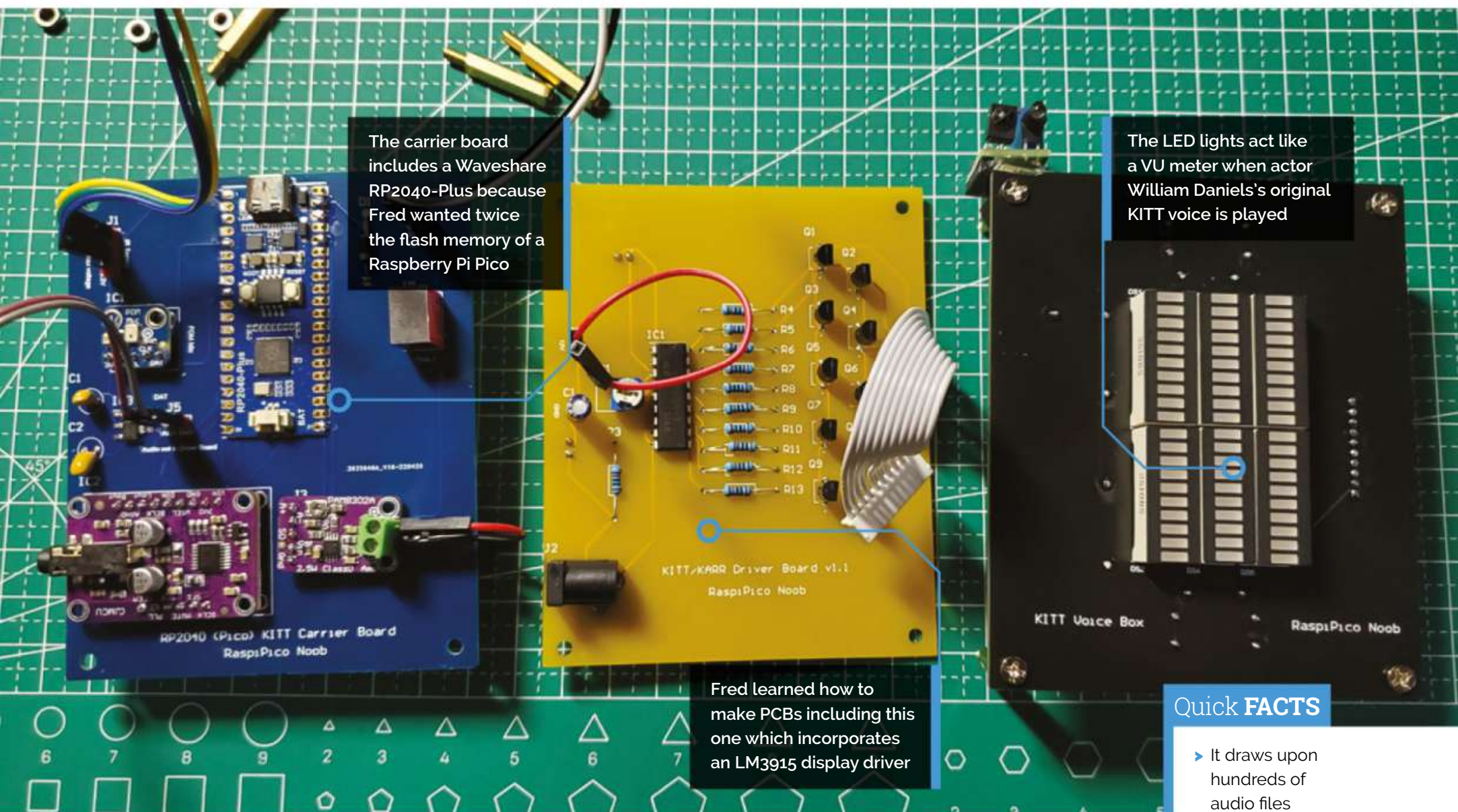
There were three main considerations: Fred needed his KITT to look, sound, and respond like the original. "Basically this project works by responding to movement using an IR sensor and then listening for sound levels using a PDM microphone," Fred explains. "Based on how loudly you speak to KITT,



► This neat arrangement of boards simply slots into the 3D-printed KITT housing



▲ Here's KITT in all his glory! He activates and listens for speech when movement is sensed



Quick FACTS

- ▶ It draws upon hundreds of audio files
- ▶ Fred wrote the software in CircuitPython
- ▶ It's based around the RP2040 microcontroller
- ▶ There are custom-built PCB boards
- ▶ An evil-twin version, KARR, is in the works

he will respond with a list of responses loaded on to the flash memory.”

Fred used a Waveshare RP2040-Plus to take advantage of its 4MB of flash memory and he took the responses directly from the TV series, saving them as MP3 files. He used an LM3915 display driver to control KITT's LED bars, a DAC audio board, and a PAM8303 audio amplifier directly connected to a speaker. “Luckily, I found a schematic online that showed how to connect the LED bars to the driver, so that helped me figure out some of the connections,” Fred reveals.

Since he didn't want to solder and wire each individual LED pin to multiple resistors and transistors, realising it would be time-consuming and prone to error, he decided to create a custom PCB. It enabled the LED sound bars to better resemble those used in the car in the TV series. “I started watching lots of YouTube videos on how to design PCBs and, after hours of trial and error, I came up with workable designs for the project.”

The extra mile

The software was written in CircuitPython. “I wrote a true loop first, checking for movement with the IR sensor,” Fred says. Once movement is detected, KITT will listen for sound. If there is none but

the IR sensor is activated, KITT will randomly say something from a list to get a person's attention. If the person responds with a loud voice, KITT will tell a joke.

“If the person responds with a loud voice, KITT will tell a joke”

“If you ask a yes/no question in a normal voice, KITT will give you a yes/no answer and it's different every time. By speaking softly, KITT says ‘you're welcome’ in a couple of different ways. All of these responses are in separate lists and they can be edited to make it say other things. I have a folder with 380 possible responses.”

In the future, he wants to add more LEDs that will light up with sound effects from the show and place in a housing similar to that seen on TV. But would he put it in his car? “It would be possible to incorporate into a car,” he says. “I would probably need to make some changes to the code to compensate for the noise levels in a vehicle, but the project already runs on 12V, so it wouldn't be a problem connecting it to an internal power source.”



▲ Fred is already working on a version that looks even more faithful to the original KITT interface

Macintosh SE/30 Raspberry Pi

Despite being faced with an empty case, Dan Beimborn managed to bring a Macintosh SE/30 back to life, as **David Crookes** explains



Dan Beimborn

Dan Beimborn has been working professionally with UNIX and Linux for the last 30 years, and has always had a soft spot for the first computers he learned on.

magpi.cc/se30

The Macintosh SE/30 was unveiled in January 1989, and it's considered by Mac aficionados to be among the best of Apple's long line of computers. Powered by a Motorola 68030 processor, and the first compact Mac to boast a high-density 1.44MB disk drive, it was a smart-looking machine with a 9-inch display but, at \$4,369, it sure was pricey.

Looking at Dan Beimborn's Mac SE/30, you'd think it was worth the outlay. Photographs suggest it's running fine 33 years on, but that's not actually true in this case. Instead, this model has been stripped of all its innards and it is running entirely on a Raspberry Pi 4 computer.

"I thought it would be relatively straightforward to use an old Mac case to fit a screen and the various Raspberry Pi components to get things working, so I set off on many long evenings

searching for parts," says Dan, motivated by nostalgic memories of playing Risk on a neighbour's Mac during study breaks in the 1990s.

Shelling out

Dan found an empty Mac SE/30 shell on eBay, buying it for a good price. He already had a Raspberry Pi 4 8GB model in an Argon One M.2 case, but he added a 1TB SSD, made use of an 18W power supply, incorporated a USB hub, bought a USB-powered speaker set, and went hunting for a suitable screen, coming across a 9.7-inch 2048×1536 4:3 LCD display by LG on AliExpress.

"The biggest challenge was fitting the screen to the bezel," he says. "I used a mini Dremel to remove some plastic where the monitor bracket



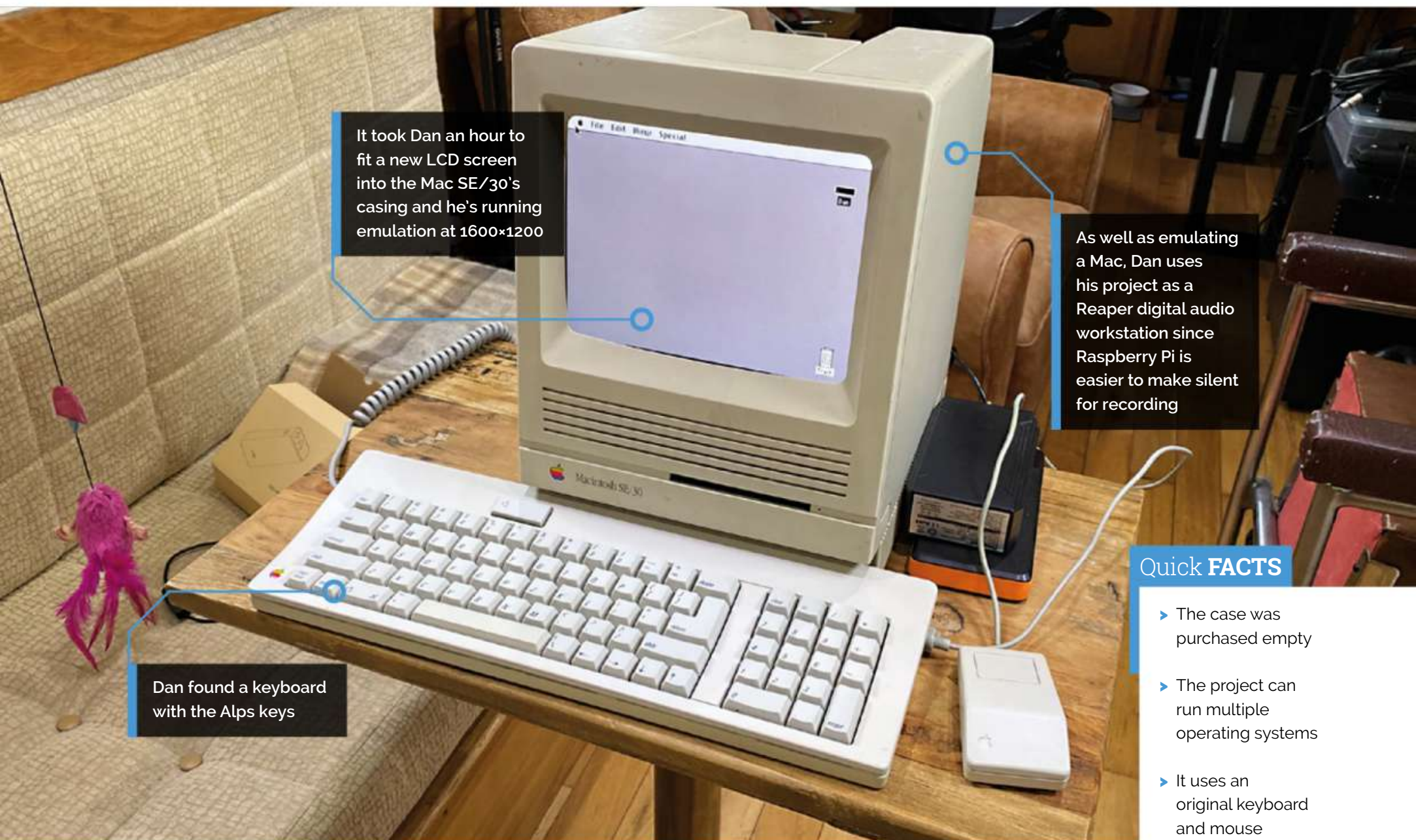
Warning!
CRT

Be careful with projects involving old television and CRT equipment. Opening up a CRT can be dangerous, risking electric shock even if the TV is not plugged in.

magpi.cc/crt

▶ The components are simply connected together using cables with two USB speakers sending sound through the sides





“ The biggest challenge was fitting the screen to the bezel ”

connected and this left me with four corners with C-shaped brackets. I carefully picked an LCD panel with the right dimensions and made tiny adjustments until it was snug. It's held in place with a few tiny screws directly into the plastic where it can't be seen, and a pair of repurposed L-brackets helped hold the screen's controller board in place.”

From that point, the main task was cable management, although Dan decided he wanted to connect an original Mac keyboard and mouse. He used an Apple Desktop Bus to USB keyboard/mouse adapter made by tinkerBOY. “It was literally plug and play, but I'll probably remap some keys.”

Sixes and sevens

The Mac SE/30 ran up to Mac OS 7, so Dan used Alex Goldcheidt's BerryBoot boot manager to select between Raspberry Pi OS, Kodi, RetroPie, and

Xubuntu. “I have Xubuntu running TwisterOS and, from here, I can start Mini vMac or MAME to run Mac OS 6 or 7,” Dan explains.

Emulation proved to be a little fiddly, however. “At first, I was having a heck of a time getting 512x384 to full-screen properly. Mini vMac has a nice screen doubling filter built in, but full screen won't automatically stretch to fit perfectly. My LCD doesn't like very low resolutions, but 1600x1200 works quite well.”

It took some research to get MAME working with Mac emulation. “Apple released the installation media for the operating system some time ago and it seems to have dropped off its website, but **archive.org** had a mirror,” Dan continues. “I was also able to make a 200MB hard drive image using MAME software, and I could then use images from other emulators to copy files back and forth, eventually getting the programs I wanted on to my hard drive image.”

Dan's certainly pleased with how it's turned out, but he's not finished with Macs yet. He now wants to restore one to its original glory. “I've been inspired by this project,” he says. “And I want to get a real CRT one going next.”



▲ The keyboard and mouse were originally yellowed, but Dan took them apart and used a home-made chemical mixture to whiten them

BirdNET-Pi

bird identification listening station

Curiosity about local wildlife led to the creation of BirdNET-Pi, which identifies birds by their song. **Rosie Hattersley** quizzes its founder



MAKER

Patrick McGuire

Patrick McGuire is a 'nature-loving tinker', citizen scientist and accidental tech enthusiast.

magpi.cc/birdnetpigitt

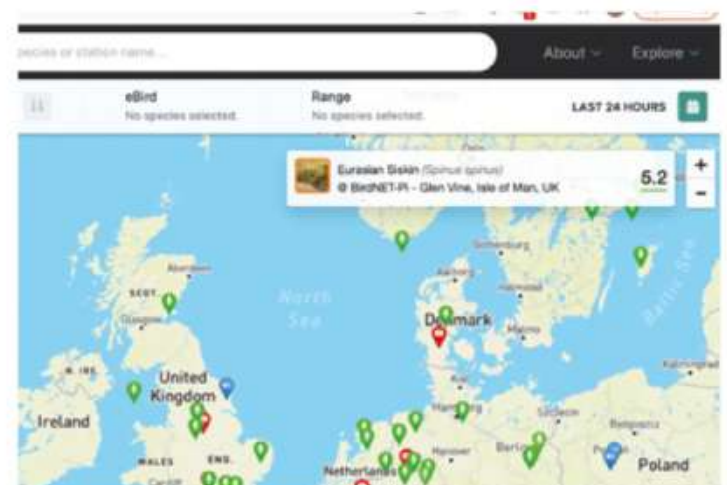
Patrick McGuire describes his BirdNET-Pi acoustic monitoring project – making use of Raspberry Pi to achieve it – as a contribution to making citizen science more accessible and affordable. The approach also chimes with his own route to discovering the open-source community: he first became interested in tech when tasked with fixing his partner’s seemingly ‘dead’ MacBook which was long out of warranty. Having successfully revived the laptop by taking a punt on installing Linux (with the wealth of useful information about it online, he reasoned it couldn’t be too problematic to set up and use; he laughs at this assumption, but the ‘dead’ MacBook is still in use), Patrick began exploring more computing projects after being given a Raspberry Pi as a present. Since then, he’s been using Linux and Raspberry Pi to “solve my problems, satisfy my curiosities, connect me with others, and contribute to citizen science.”

Credit where it’s due

BirdNET-Pi is based on the Lite version of the eponymous BirdNET research platform created by Stefan Kahl and others at the K Lisa Yang Center for Conservation Bioacoustics at the Cornell Lab of Ornithology and the Chair of Media Informatics

at Chemnitz University of Technology. Their research is mainly focused on the detection and classification of avian sounds using machine learning, and works on most computing platforms, as well as having iOS and Android apps. The BirdNET database recognises around 3000 species of birds, around 1000 of which are European species.

Patrick’s version, BirdNET-Pi, expands the user base for the acoustic monitoring platform, using Raspberry Pi OS for on-device analysis and processing. He thinks the way “BirdNET-Pi automatically creates clips of detected bird songs with spectrograms to visualise the audio





BirdNET-Pi works with almost any USB microphone and sound card, automatically recording birds singing

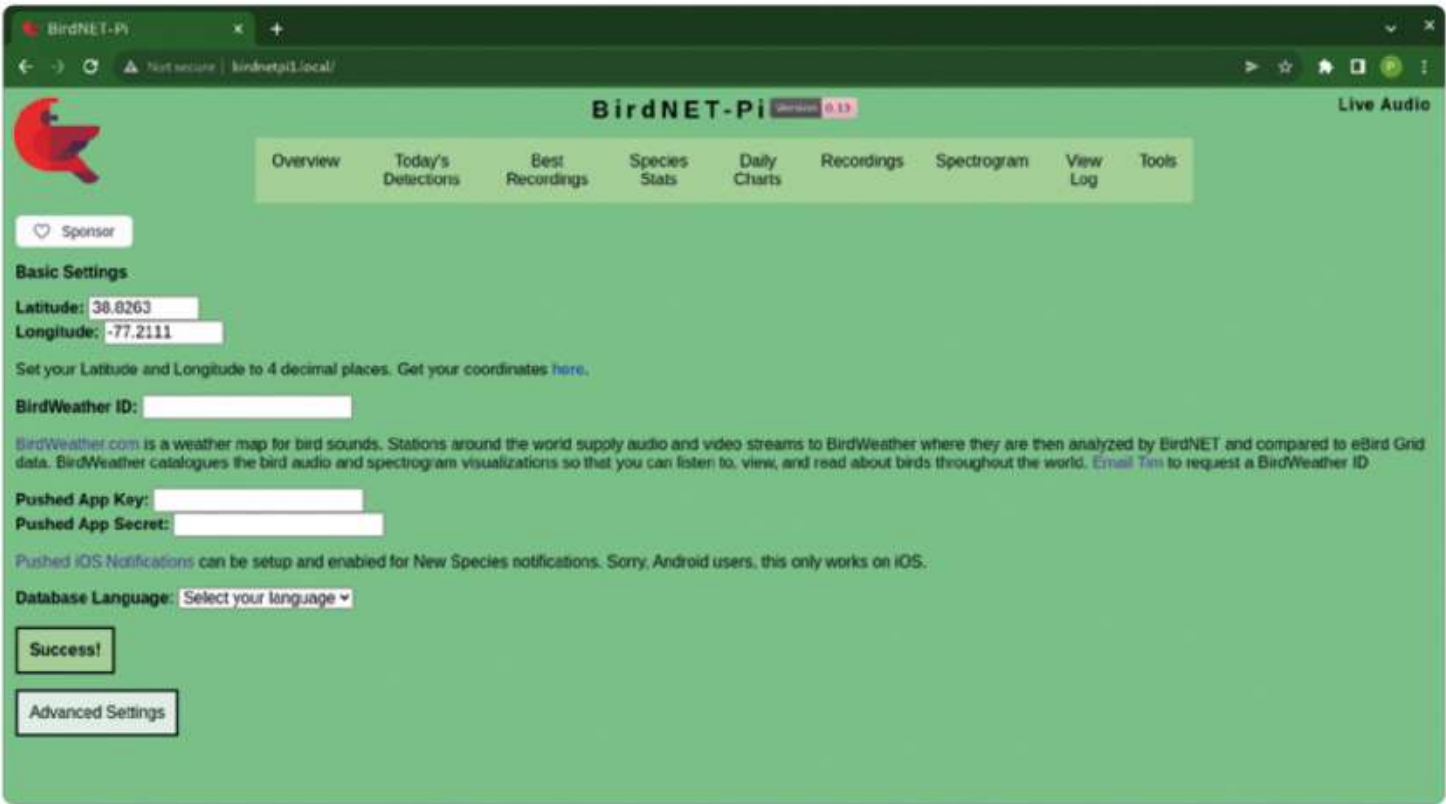
Raspberry Pi running BirdNET-Lite analyses bird sounds, checks them locally against its database, and logs the species' presence

An Ethernet cable running from the acoustic listening station means it can be positioned in a wildlife-rich location, but bird calls can be reliably logged and details sent to BirdNET

Quick **FACTS**

- ▶ BirdNET-Pi began as a means to impress Patrick's partner's mother
- ▶ The logo is a northern cardinal, the official bird of Virginia
- ▶ BirdNET-Pi is built on the TFLite version of BirdNET
- ▶ BirdNET-Pi recently caught the imagination of UCL...
- ▶ ...which ran a sold-out workshop on it in June

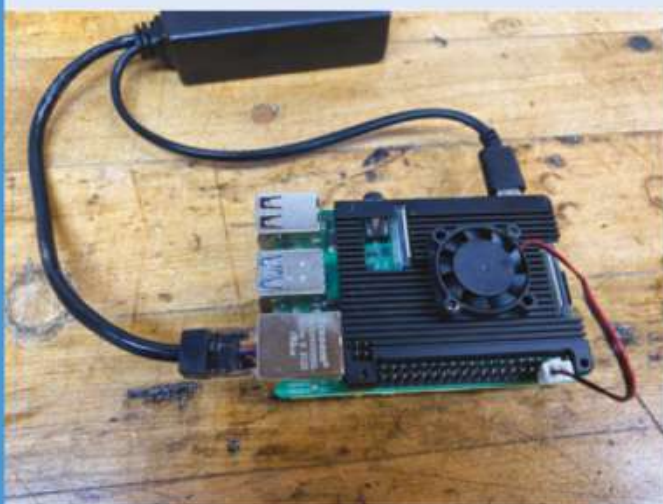
- ▶ You can specify the listening station's exact location among other settings
- ▶ The BirdWeather map shows species recently detected by BirdNET-Pi and other acoustic monitors



Identify bird sounds



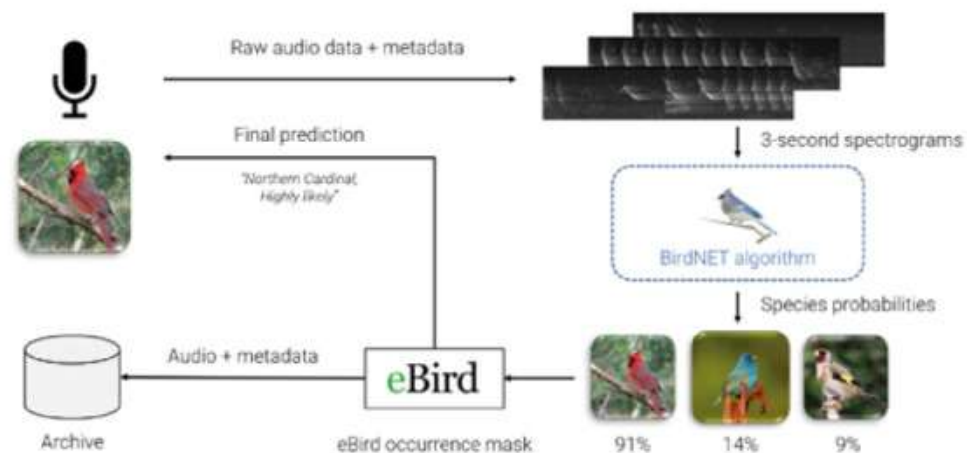
01 A waterproof housing, Raspberry Pi, microphone, fan, and cables make up the basics of an in-the-field BirdNET-Pi station. Photos from Bill Powers' setup (magpi.cc/birdnetbuild).



02 The Power over Ethernet card and PoE adapter, plus a power supply with fan, ensure BirdNET-Pi can be used to remotely record and identify bird species.



03 The dashboard view for BirdNET-Pi shows a bird call has been recorded, analysed, and checked against the songbird database, revealing that a pileated woodpecker has been detected.



▲ BirdNET uses sophisticated AI algorithms to predict which bird is singing

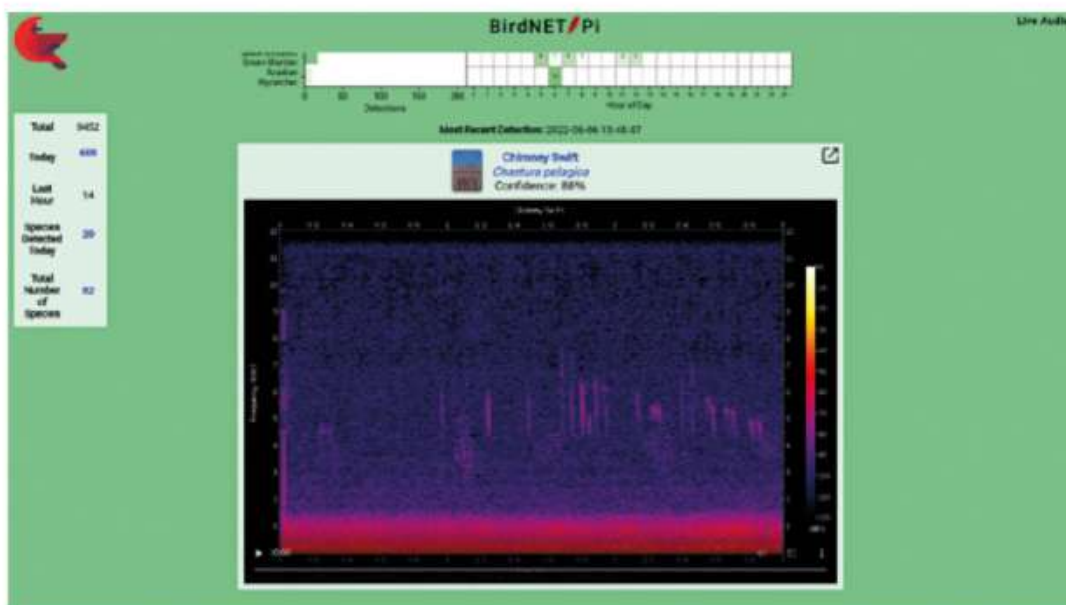
and each bird's unique vocal signature" is a game-changing feature.

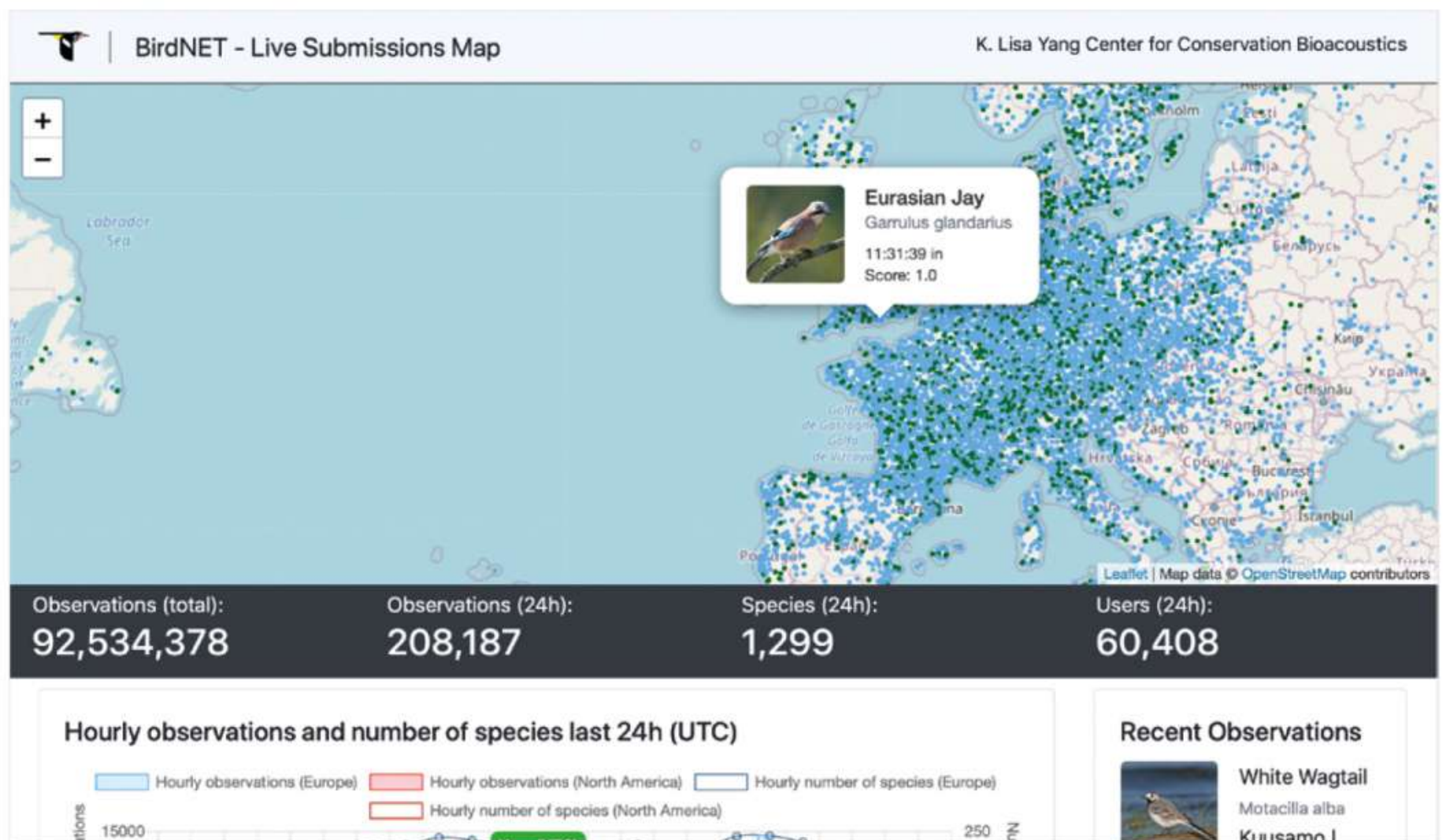
The original BirdNET was trained in the Sapsucker Woods close to the Cornell Conservation Bioacoustics Labs. BirdNET-Pi was developed in Virginia, where Patrick lives, and is one of dozens of listening stations that have been set up and left in situ with a wireless connection allowing for remote bird monitoring. "Since all of the analysis is performed on the device itself, researchers can deploy their BirdNET-Pis in places of avian interest where human monitoring for the same duration would be impossible – they can come back later to retrieve the device and all of its analysed data."

BirdNET-Pi also uses Icecast (icecast.org) for live audio-streaming, so fellow nature watchers can tune in and hear recordings of live birdsong from other BirdNET devices. There are a handful of broadcasting stations in the UK, plus dozens of other sites you can visit virtually to hear bird song recordings and learn about the species.

Based on the BirdNET-Lite bird sound analysis model, it works on a Raspberry Pi 3B+, 4, or

▼ Recording of a successfully identified chimney swift singing





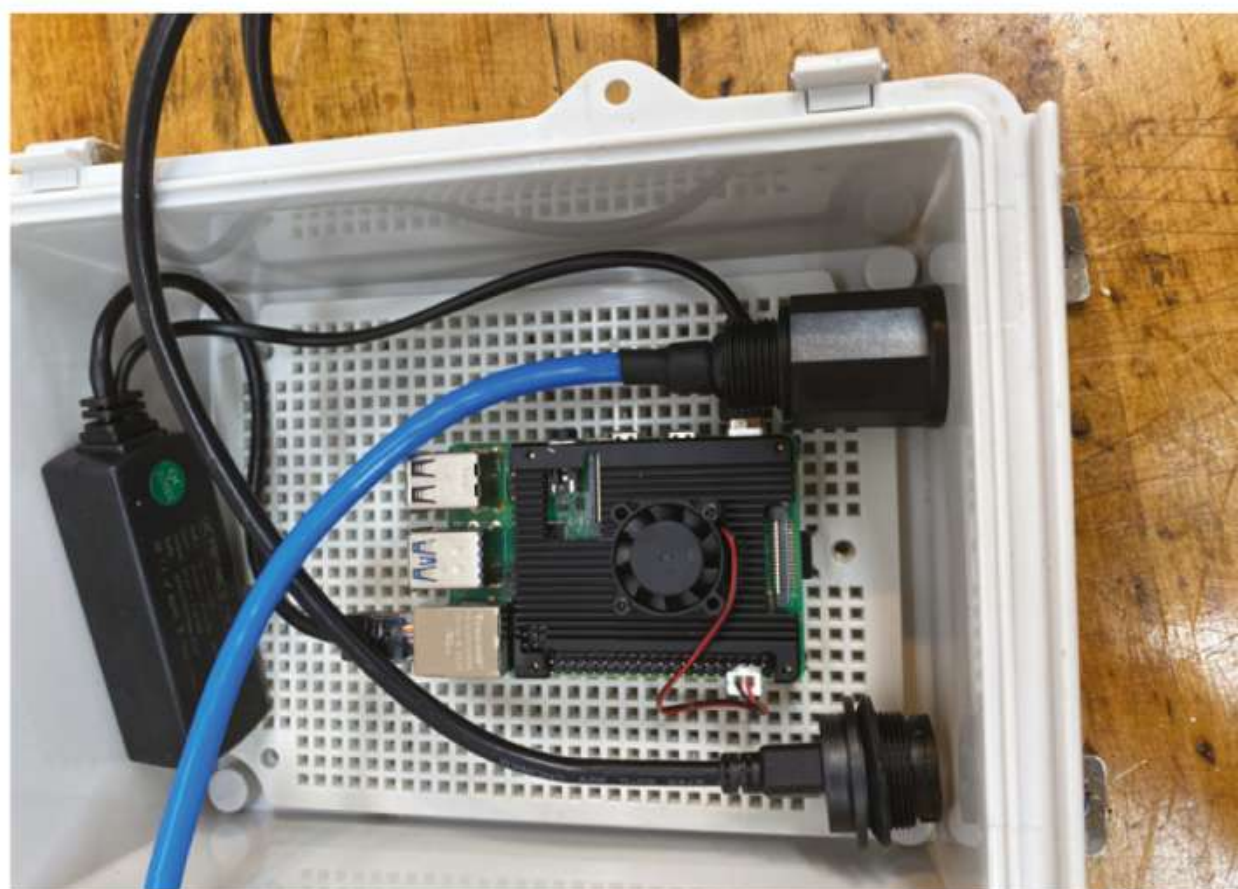
Zero 2 W, requiring the 64-bit version of Raspberry Pi OS. Raspberry Pi was chosen for its affordability, and Patrick is particularly proud of the fact BirdNET-Pi works with such a range of USB sound cards and microphones. “This allows users to use hardware that fits their needs and their budget, lowering the barrier of entry for participation.”

Raspberry Pi proved a great choice. “The original version, BirdNET-system, only ran on x86 machines. It could record up to four hours each day and spent the other 20 hours of the day analysing the data! When the project migrated over to using the BirdNET-Lite model, I was able to port everything to the AARCH64 architecture and BirdNET-Pi was born.”

Global reach

Although BirdNET (and BirdNET-Pi) was developed and AI trained in the US, it’s now being used by ornithologists and nature lovers to identify species across the globe, which means the database of confirmed bird sightings and sounds has broadened significantly. Patrick lists 18 countries in which wildlife enthusiasts are actively using BirdNET-Pi, from New Zealand to Romania, to South Africa, Canada, and Sweden, as well as the UK, Germany, and USA. Along with the iOS and Android app versions of BirdNET, this Linux-based Raspberry Pi version serves to expand the geographical reach of the bird ID project.

“It’s now being used by ornithologists and nature lovers to identify species across the globe”



▲ A weatherproof box protects Raspberry Pi out in the field

Night camera

How well do you sleep? **Rob Zwetsloot** takes a look at a sleep monitoring system beyond his wildest dreams



MAKER

Emilio Viudez Ruido

A trained mechanical engineer who ended up working as a business analyst in the energy trading industry. 'Making' has become his antidote against too many PowerPoint slides.

Do you use apps to track your sleep? They're quite interesting, even if they're not always easy to understand. Emilio Viudez Ruido is taking a different approach to vibrations and snore recording: he's taking photos of himself to try and find some patterns.

"I use a Raspberry Pi Zero with an infrared camera to capture the pictures at 15-second intervals," Emilio tells us. "Some of them are used to train an image classifier to differentiate three different poses: sleeping on my left side, on my right side, or on my back.

"Once the model is trained, I use it every morning to determine the sequence of poses from the 16,700 images collected during the night. With these series, I can calculate some statistics which I hope would reveal some interesting patterns. Additionally, I'll combine this information with my subjective assessment of readiness, and with



▲ An example of the kind of photo Emilio gets back

the objective assessments from my Oura ring (a commercial sleep tracking device)."

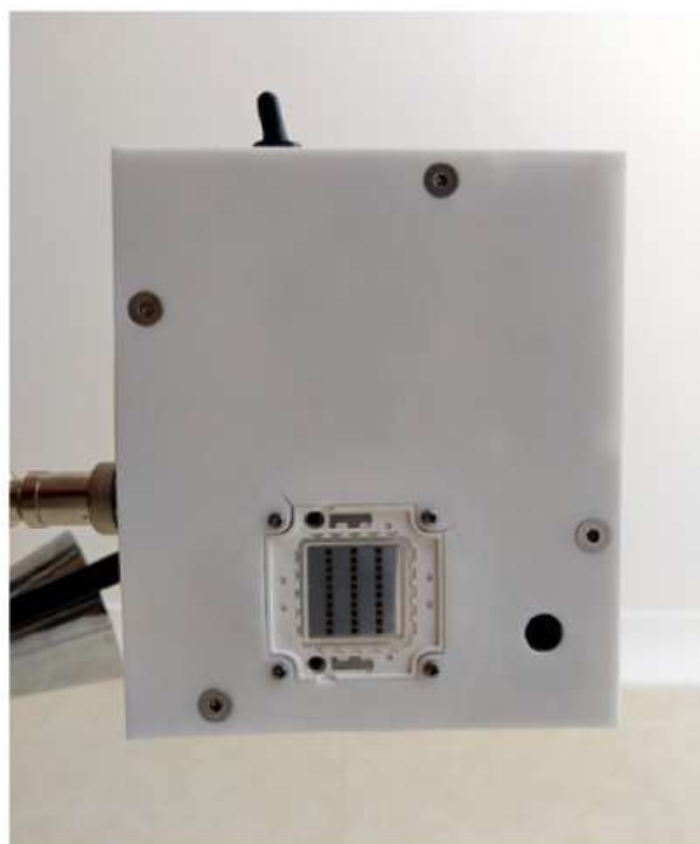
While Emilio says he's in good health, he feels his sleep quality has degraded over the years: "I have been trying to understand why and what could I do to improve it. There is nothing better than waking up in the morning feeling fully recovered and energised!"

Bedroom eyes

The concept for the tracker came from reading *The MagPi*, specifically a nature-watching camera project that used the NoIR Camera Module.

"I purchased a sleep tracker which provided me with very useful feedback," Emilio explains. "However, sometimes the results from this device would not necessarily match how I was feeling in the morning. Some time afterwards, while reading in a *The MagPi* issue a project where the NoIR Camera Module was used to take pictures of animals at night, I realised I could become one of those 'animals', and see if my own footage could help me understand those gaps."

The software side used a series of images that Emilio had manually labelled as back, left, and right for the poses he was sleeping in. He then fed



▶ Emilio specifically made the box simple as it's purely for utility – you can't see it when it's working!

them into Google's Teachable Machine to train the model to recognise the poses he wanted.

"This is a great way to very quickly sense how feasible the whole project would be," Emilio says. "With 100 labelled samples for each class, and in less than ten minutes, I had a trained model downloaded in my local machine running inferences on the entire set of captured images."

He then 3D-printed a cage for the components after some prototyping, including infrared LEDs to help with lighting. It's powered from the mains, once it became clear that even beefy rechargeable batteries would not quite suffice.

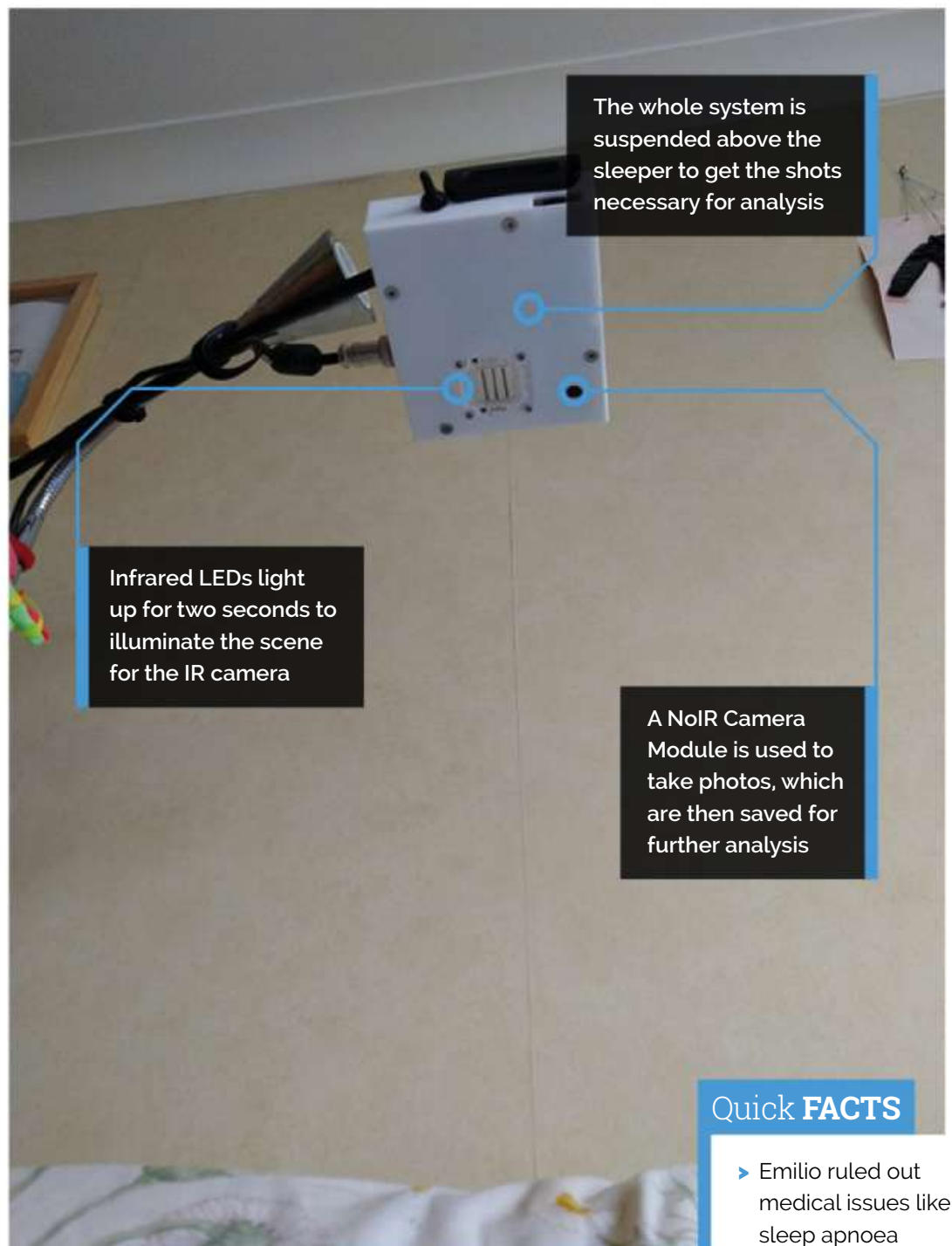
“ This is a great way to very quickly sense how feasible the whole project would be **”**

Dream warrior

During sleep, a Python script runs over nine hours and takes one image every 15 seconds, which Emilio can then analyse.

"After manual inspection of all pictures over several nights, I can confirm the 15-second cadence is fit for purpose. I can even see some REM (rapid eye movement)," Emilio says. "So far, I only have some basic statistics from the last ten nights: on average I sleep 27% on my back, 41% on my left side, and 33% on my right side. Another interesting observation is that eyeballing the Oura ring sleep score against the percentages on each night, I can see that high scores tend to occur on nights with more 'on right' sleep."

Emilio is continuing to work on the system, increasing accuracy, doing sleep research, and searching for patterns in his own sleep. **M**



Quick **FACTS**

- ▶ Emilio ruled out medical issues like sleep apnoea
- ▶ Identifying which side he was sleeping accurately took more training
- ▶ Emilio plans to add more things for the system to measure
- ▶ Real-time analysis using an Edge TPU will occur eventually
- ▶ The limited LED use is so they don't overheat



▲ A lot of components are stuffed into the box. Emilio enjoyed figuring out the circuitry to control the IR LEDs

SUBSCRIBE TODAY FOR JUST £10

Get 3 issues + FREE Pico W



Subscriber Benefits

- ▶ **FREE Delivery**
Get it fast and for FREE
- ▶ **Exclusive Offers**
Great gifts, offers, and discounts
- ▶ **Great Savings**
Save up to 35% compared to stores

Subscribe for £10

- ▶ Free Pico W
- ▶ 3 issues of The MagPi
- ▶ Free delivery to your door
- ▶ £10 (UK only)

Subscribe for 12 Months

- ▶ Free Pico W
 - ▶ 12 issues of The MagPi
 - ▶ Free delivery to your door
- | | |
|----------|---------------------|
| £55 (UK) | £90 (USA) |
| £80 (EU) | £90 (Rest of World) |

☎ Subscribe by phone: **01293 312193**

📧 Subscribe online: **magpi.cc/subscribe**

✉ Email: **magpi@subscriptionhelpline.co.uk**

Subscribe for £10 is a UK-only offer. The subscription will renew at £30 every six months unless cancelled. A free Pico W is included with a 12-month subscription in USA, Europe and Rest of World. (No Raspberry Pi Pico W with £5 Rolling Monthly Subscription).

SUBSCRIBE TODAY AND GET A **FREE** Raspberry Pi Pico W

Subscribe in print
today and get a **FREE**
development board

- ▶ A brand new RP2040-based Raspberry Pi Pico W development board
- ▶ Learn to code with electronics and build your own projects
- ▶ Make your own home automation projects, handheld consoles, tiny robots, and much, much more



This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.

 Buy now: magpi.cc/subscribe



SUBSCRIBE on app stores

From £2.29

 Available on the
App Store

 GET IT ON
Google Play

SUCCESS STORY magpi.cc/success

Korg

Expensive instruments exclude aspiring musicians, so Korg set about making its synths much more accessible. By **Rosie Hattersley**

Electronic instrument maker Korg (korg.com) began life in Japan in the 1960s, bringing to market first drum machines, then Japan's first synthesizer, and the world's first needle-type tuner. It came to international prominence with the rise of electronic music in the 1970s and 1980s. Demands for ever more capable synthesizers – as well as more affordable models – saw Korg expand its product lines into both home hobbyist and professional performer arenas. Keen to capitalise on the exciting possibilities of electronic music production, which was developing at a rapid pace alongside the exponential growth of home and office computing, Korg took an early interest in DSP-based synthesizers.

In the 1990s, this meant using custom-built DSPs (digital signal processors) along with off-the-shelf parts from Motorola and Texas Instruments. In 2005, Korg started to use Linux running on Intel processors as the 'DSP' for its high-end keyboards, starting with the Pentium-based OASYS, followed by the Atom-based Kronos. Most recently, Korg has launched a new line of more accessible digital synthesizers,

including the wavestate, modwave, and opsix, which take advantage of Raspberry Pi Compute Module 3's processing abilities and cost a relatively affordable \$799.

In an interview (magpi.cc/korgkeys) in November 2021, you can hear from Korg R&D in California, the team that developed the wavestate and modwave: General Manager Andy Leary, and Manager of Product Development Dan Phillips.

THE CHALLENGE

Modern synthesizers need a lot of horsepower to deliver professional-quality audio, feature-richness, and high polyphony (the number of voices that can be played simultaneously). Classic Korg instruments used multiple custom ASICs (Application Specific Integrated Circuits) for the task. The flexibility of these ASIC-based systems was necessarily limited since the basic functionality was baked into the hardware design. For example, a given set of ASICs would provide a single synthesis algorithm (such as sample playback) with a fixed number of voices. These bespoke systems also didn't come cheap; 1988's M1 keyboard was listed for \$2749, equivalent to about \$6800 in 2022 dollars.

To break free from the constraints of ASIC designs, Korg's California-based Korg R&D team started to work on DSP-based synthesis. The idea was that the hardware would be generalised, with the bulk of the features dependent on software rather than hard-wired electronic circuitry. Such a system would be capable of running many different types of synthesis algorithms, such as physically modelled acoustic instruments, virtual analogue synths, sample playback, tonewheel organs, and so on.

Korg R&D started by designing a custom DSP





built for usage in arrays, with a cluster of DSPs communicating with one other. After producing a prototype based on this hardware, they switched to using an array of Motorola DSPs for their OASYS PCI, a PCI-based synthesis and effects system for Macs and Windows computers which shipped in 1999. In contrast with standard ASIC-based instruments,

“ The original Wavestation had legendary status, so news of a brand-new model was a big deal ”

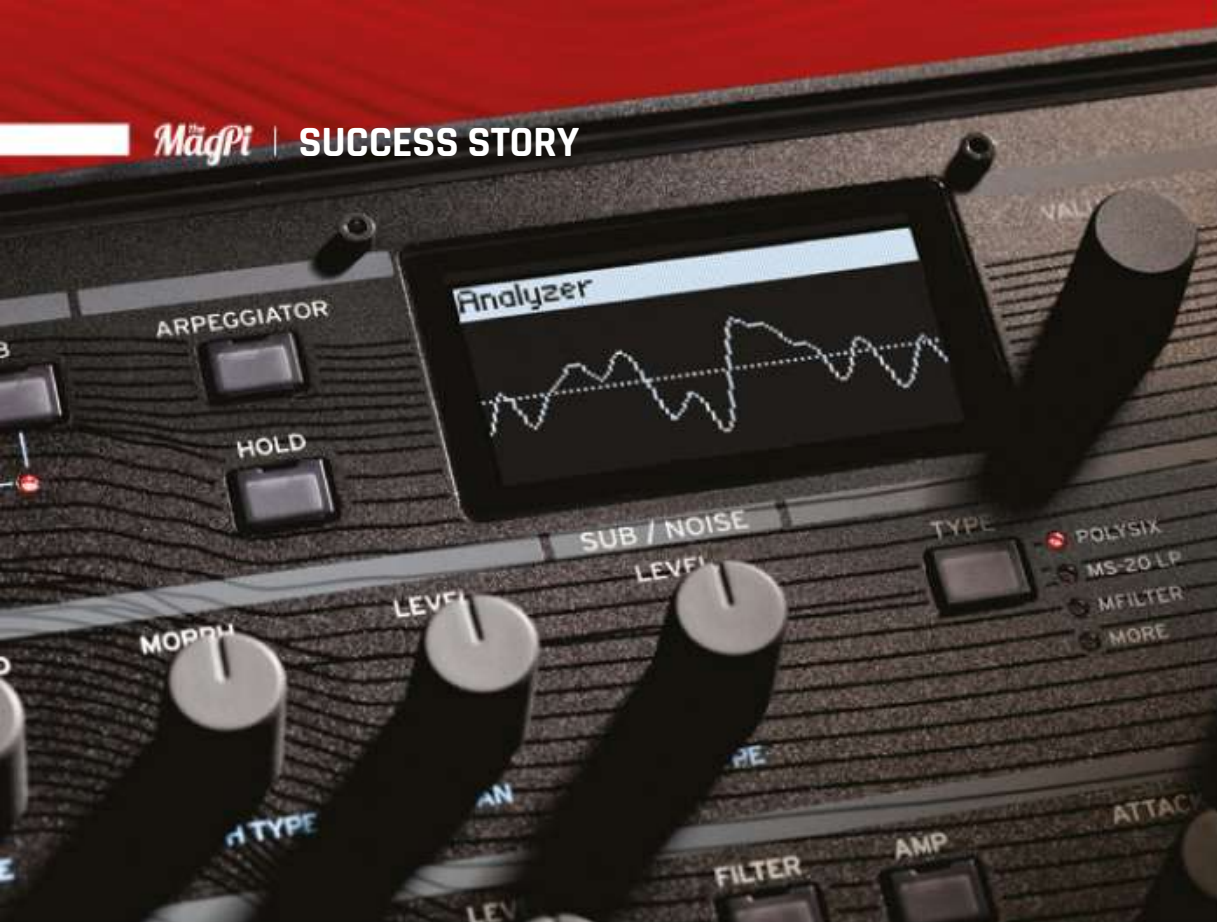
OASYS PCI shipped with over 40 synthesis algorithms and over 130 effects algorithms. Over the life of the product, third-party developers continued to expand the platform, providing many more algorithms.

As flexible as it was, OASYS PCI had various technical and commercial limitations. With five high-end DSPs in a custom design, the materials cost was still high. Each DSP had to have its own memory, which added to the cost and introduced challenges in allocating processes across the array. Users had to manually specify how many voices each synthesis algorithm would play. Following the same model as its ASIC predecessors, audio processes were separated from control processes; the former ran on the DSPs, and the latter (with some exceptions) ran on the host CPU. This

required careful handling for synchronisation of the audio and control processes and also consumed bandwidth over the PCI bus. Finally, while the amount of raw processing power was high for a DSP platform, it had lower polyphony than the ASIC-based systems.

To improve these factors, Korg R&D's next product was designed around Linux running on an Intel Pentium processor. Released in 2005, the OASYS was a 'workstation' keyboard, meaning that it combined synthesis, effects, sampling, live performance features, and audio and MIDI recording; its interface included both an articulating touchscreen and banks of sliders and knobs. The Pentium provided all functionality, including the user interface, operating system, and audio processing. For the latter it was treated as a monolithic DSP, eliminating the DSP array's issues of inter-chip communication and multiple banks of memory. Control and audio processes were also unified and synchronised, eliminating the issue of bandwidth constraints. Unlike its predecessor, it could dynamically allocate processing power to different synthesis algorithms, responding in real-time to the notes being played. The combination of a relatively fast CPU, a real-time kernel, and tight assembly coding meant that, despite providing a set of higher-fidelity synthesis algorithms, its polyphony was significantly greater than its ASIC counterparts.

Designed as a high-end instrument for professional users, the OASYS was large, heavy, and expensive, coming in at \$8000 for the 88-key,



piano-action model. “The Oasys was a bit of a shock to many of our users because it was more than double the cost of our previous machine,” notes Andy Leary. Still, “It made some incredible sounds that other things just couldn’t do. It really was a flagship instrument, and kind of a groundbreaking instrument.”

The Kronos, released in 2011, continued to build on the OASYS technology while achieving a more standard price point. A switch to the Intel Atom quad-core processor helped to reduce costs, and various software-based improvements expanded its appeal. They’d clearly struck the right note: the Kronos was a strong seller for ten years, with a few tweaks along the way. It was succeeded by the Nautilus, based on the same technology, in 2021.

Even as the Kronos was dominating arena shows and recording studios, Korg R&D was busy working on the next iteration of its synthesis platform. This time, the goal was to hit lower price points while not compromising on features and fidelity. Had Intel continued down the route of more powerful processors that became progressively less expensive, they might have been a viable option, but the Pentium chip maker was very focused on computers and laptops. Instead, Korg eventually chose TI’s OMAP platform, which combined an ARM CPU with a DSP, for 2017’s Grandstage and Vox Continental products. Despite having the CPU and DSP on the same die, it still had some of the same issues as the older OASYS PCI. “It still wasn’t one chip that was doing all the work. We had to deal with this kind of interconnection issue between the DSP part, and the main CPU that was running the user interface,” explains Andy.

THE SOLUTION

For their next product, Korg R&D’s goal was to make products accessible to all musicians, by reaching the sub-\$1000 price point. Eventually, they realised that solutions designed for computers “just cost a little too much”. They switched to

Raspberry Pi “and basically get all of what we needed, for a lot less. It’s smaller, cheaper, faster, lighter, better, all that stuff. It was a clear path for us,” says Dan Phillips. Another compelling aspect was that with Compute Module everything was ready to go. Korg could focus on the custom aspects of their product, such as professional-quality audio hardware, the physical keyboard, and the extensive physical control surfaces, and then just plug in a single part to provide the CPU, RAM, and storage. “That part of the work is already done. It’s like any other component; we don’t have to lay out the board, build it, and test it.”

WHY RASPBERRY PI?

Korg was also persuaded by Raspberry Pi’s commitment to a long-term roadmap producing and supporting its products – a key appeal for Korg, who had occasionally been forced to accommodate

“ Not everyone understands that Raspberry Pi is actually making the sound ”

abrupt changes due to the discontinuation of DSPs, memory, and other components.

“It made sense to go with a company that was making something in huge volumes and was committed both to continued production and to pushing the technology forward into the next generation. That reassurance is exactly what a business needs,” explains Dan.

It also helped that Korg didn’t need to do that much to take advantage of it. In fact, they bought several Compute Module 3 units, tried them out and realised “hey, we could make this work,” says Andy.

THE RESULTS

In early 2020, Korg R&D announced the wavestate, a successor to its 30-year-old Wavestation and its first instrument to use the Raspberry Pi Compute Module 3. The original Wavestation had legendary status, so news of a brand-new model was a big deal and expectations considerable.

The Wavestation used ‘Wave Sequencing’ to crossfade between samples, combining them into new sounds. The wavestate builds on this with ‘Wave Sequencing 2.0’, which introduces extensive real-time control, sophisticated pattern

creation and manipulation inspired by 20th-century algorithmic composition, and controlled randomisation. It also has significantly higher audio quality and a much more powerful synthesis architecture than the original, while offering twice the polyphony.

The Korg R&D team is based in California and has been working together for many years, notably on the OASYS and Kronos, before turning their hands to the wavestate and its first follow-up, the modwave wavetable synthesizer.

The team uses software to prototype their instruments before implementing hardware designs. With the basic software platform already functional, developing the wavestate using the Compute Module 3 took a fairly modest year from inception to its late 2020 launch. The setup has two circuit boards. The main panel board contains all of the user interface elements, including display, buttons, knobs, wheels, and other synth-specific controls, along with MCU microprocessors to support them and communicate with the CM3. The other circuit board has subsystems for audio, MIDI, the musical keyboard, and power, plus the socket for the CM3.

Andy describes the setup as “Very simple. Two boards. The philosophy that we’re going with is that, when we make new products, we can keep the same main board that has the processor, audio, and so on. The front panel board provides differentiation; it can be customised as



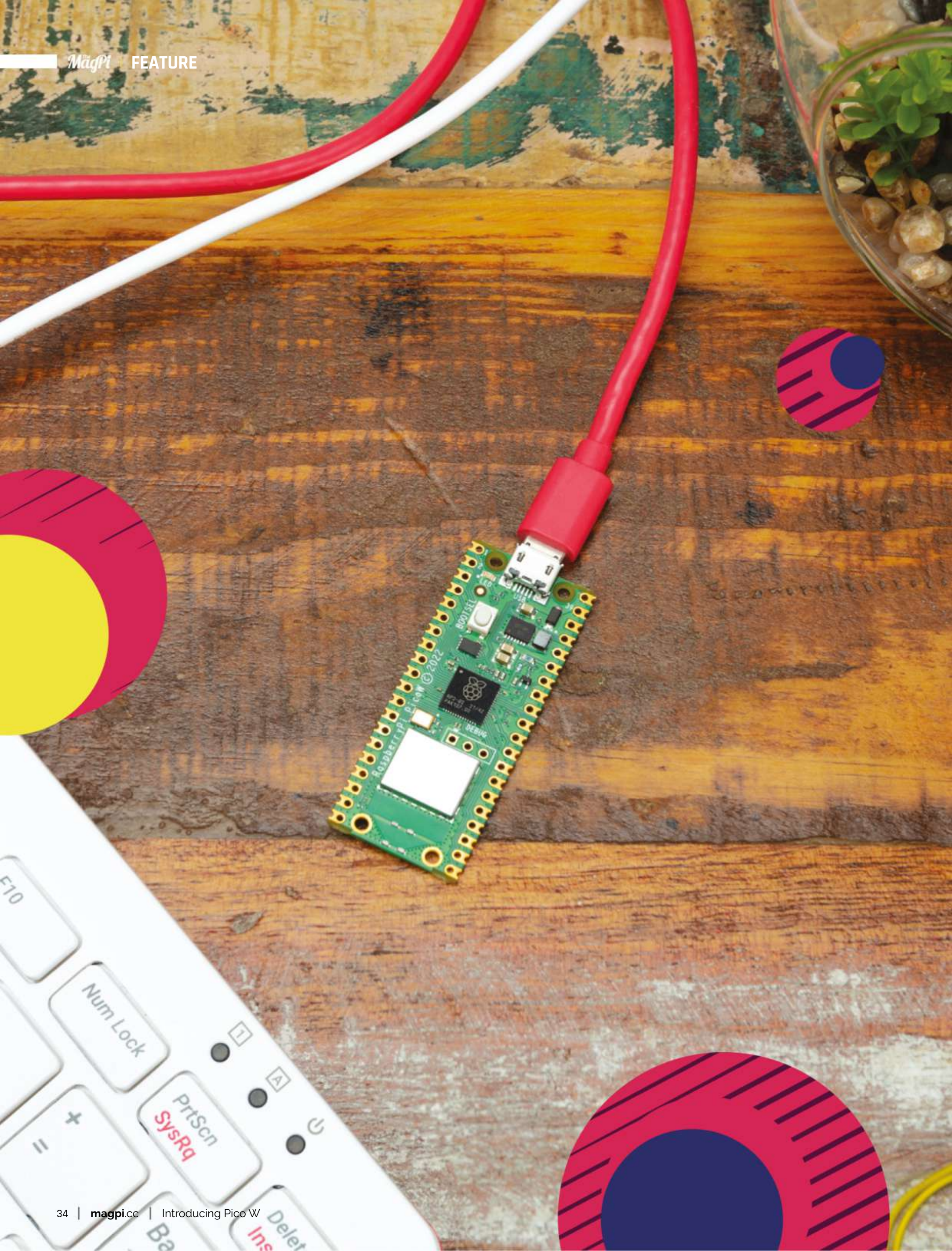
required, and tailored to the precise needs of a specific instrument.”

Dan tells us that “not everyone understands that Raspberry Pi is actually making the sound – many people assume that it’s not.” Others assume that because it uses the famously low-cost Raspberry Pi, the wavestate should be cheaper than its relatively modest \$799 – but actually, the reverse is true. “The CM3 is more cost-effective than our previous Pentium and Atom components, but it’s still significantly more expensive than our traditional DSPs or ASIC solutions. We use the CM3 because it’s very powerful, which makes it possible to create deep, compelling instruments.”

Despite a few grumbles and gripes, the wavestate has been very well-received. “It’s been a really successful product for us and has got a lot of attention. People are very excited about how much the product can do for the price point. Somehow it seems like it was a great product to launch at the beginning of a pandemic.”

Asked to quantify just how successful the move to Raspberry Pi has been, the pair report that sales have far exceeded their expectations. “Korg is privately held and does not publish sales figures,” Dan reminds us. “However, we can certainly say that these products have been very well-received by the marketplace!”





INTRODUCING

PICO W

Raspberry Pi's RP-2040 microcontroller development platform gets native wireless connectivity. By **Lucy Hattersley**

Raspberry Pi Pico has been a standout star of recent years. The tiny Pico development board contains a microcontroller chip designed by Raspberry Pi called 'RP2040'.

Programs are loaded onto Pico via a separate computer, such as a larger Raspberry Pi board, and run automatically as soon as the power is turned on. The GPIO pins provide input and output and a vibrant community and maker scene have built up to get the most from Pico.

This month, we are incredibly proud to reveal the all-new Raspberry Pi Pico W. As the 'W' moniker suggests; Pico W is packing wireless connectivity.

This tiny board now houses a silver square module containing an Infineon CYW43439 (magpi.cc/CYW43439) device. This enables RP2040 to connect to the internet via a wireless interface using an 802.11n, single-band (2.4GHz) connection. The module sits alongside an familiar trapezoidal antenna on the edge of the board. All of this required a clever engineering design, and we're going to reveal all in this feature.

Raspberry Pi Pico remains a low-cost development platform. And you can pick up

Pico W from just \$6 (around £5). It is programmed by connecting to another computer, such as a Raspberry Pi 4 or 400, or a similar Linux, macOS, or Windows computer. Programs are created using the C/C++ SDK or MicroPython.

“ All of this required a clever engineering design ”

Files are drag and drop placed onto Pico W via the USB connection or SWD pins, and you can interactively debug code running on Pico W.

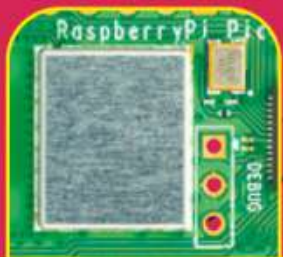
There's never been a better time to get into programming with Raspberry Pi Pico W. Pico can be used to control many hardware projects of your own design, and there is a huge ecosystem of kits and components.

We believe that wireless LAN is a game-changer for Pico. It will enable the device to connect from the edge to your network and the wider internet, and pass data back for storage and analysis.

We can't wait to see what folks make with Pico W.

GET TO KNOW

PICO W



1 WIRELESS LAN

The Infineon CYW43439 wireless chip is housed within this silver-coloured package and provides a single band of 2.4GHz wireless LAN (802.11n).



2 OSCILLATOR

The AEL1210CS oscillator provides the 12MHz clock frequency that keeps RP2040 ticking along.



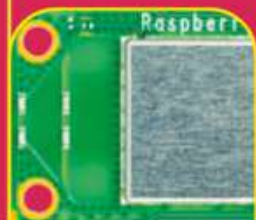
3 SPI EEPROM

This small EEPROM chip houses the code used by Pico W to perform its operations. The programs you upload to Raspberry Pi Pico are stored here.



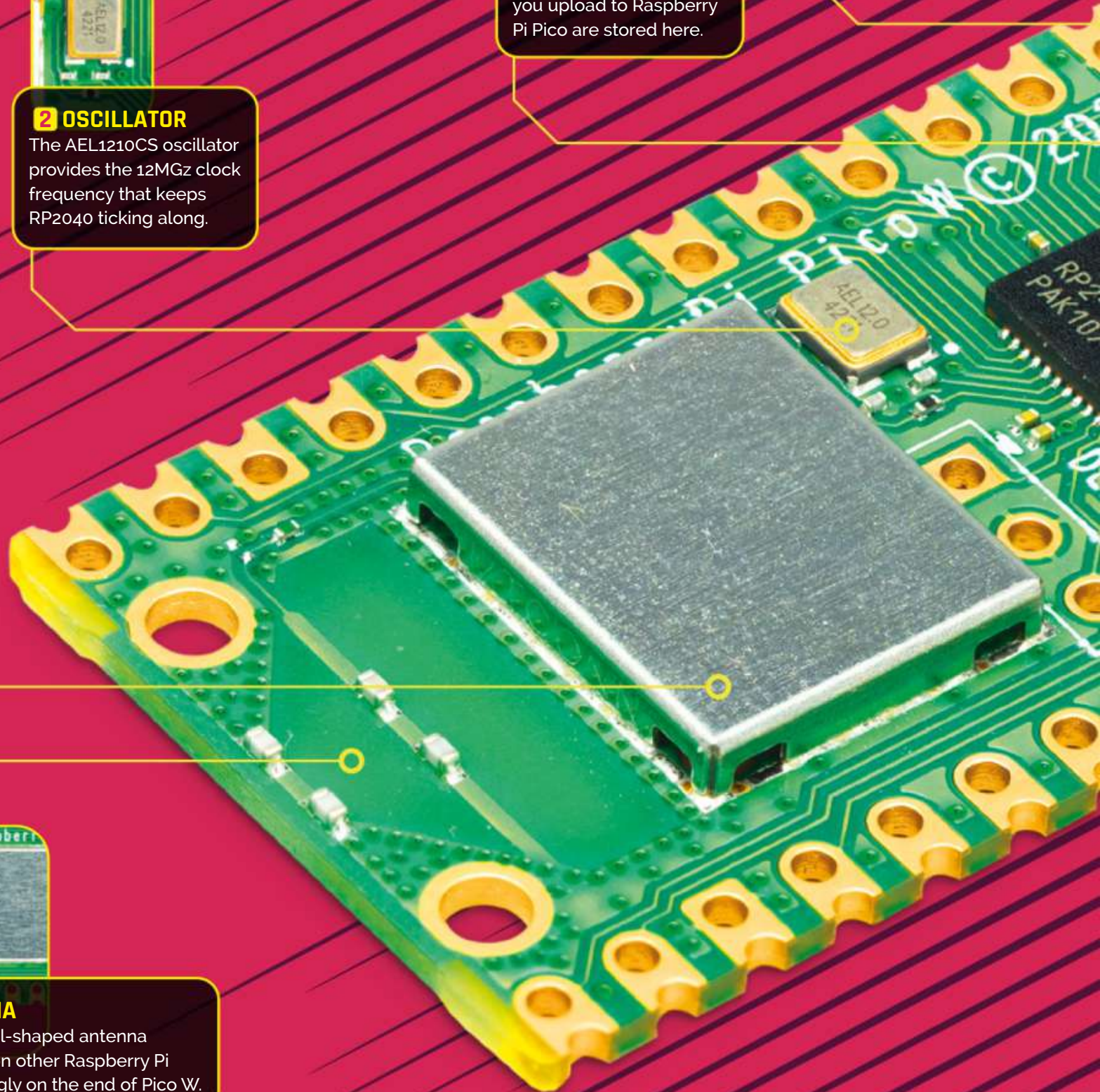
4 BOOTSEL

This button is used to connect Pico W to a computer. It then appears as a drive on the other machine.



10 ANTENNA

The trapezoidal-shaped antenna design found on other Raspberry Pi boards fits snugly on the end of Pico W.

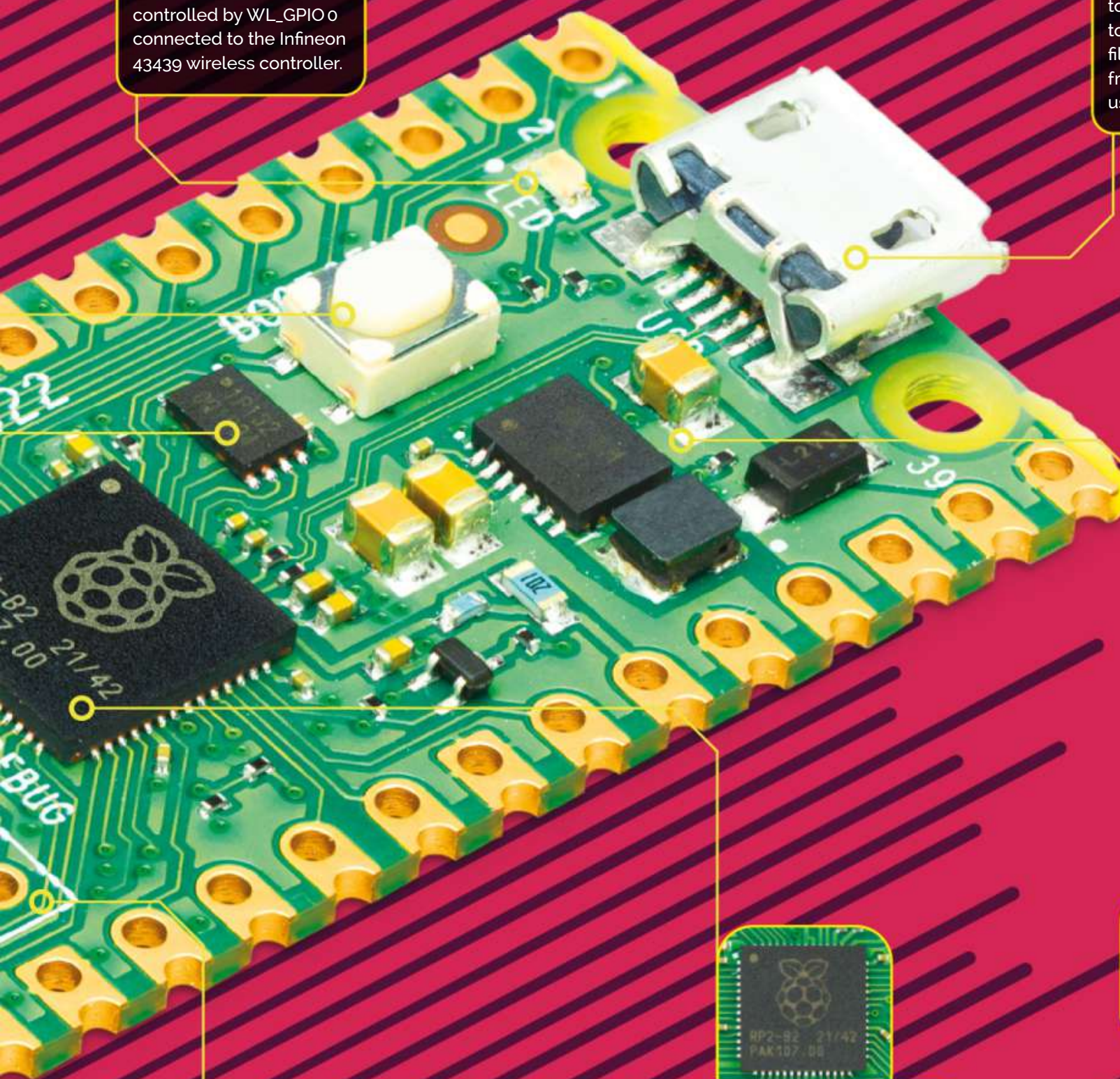


**5 LED**

A single LED remains on the board and is now controlled by WL_GPIO0 connected to the Infineon 43439 wireless controller.

**6 MICRO USB**

A single micro USB port is used to provide power and connect to another computer so code files can be transferred to and from Pico W. This can also be used for direct code debugging.

**7 POWER MANAGEMENT**

These chips form the power management for Raspberry Pi Pico, which is fixed at 3.3V.

**8 RP2040**

The beating heart of Pico is the RP2040 microcontroller, appropriately housed in the black package and adorned with a Raspberry Pi logo.

**9 DEBUG PINS**

The three debug pins have been relocated from the edge of the board to a more central location.

TAKE A LOOK
AT THE BACK OF

PICO W

1 GPIO PINS

On the two sides of Raspberry Pi Pico W are the 40 GPIO pins used to provide I/O between Pico W and other hardware. All of the GPIO pins are labelled on the rear of Pico W.

2 CASTELLATIONS

The GPIO pins are on a 21mm × 51mm 'DIP' style 1mm thick PCB, with 0.1" through-hole pins, also with edge castellations.

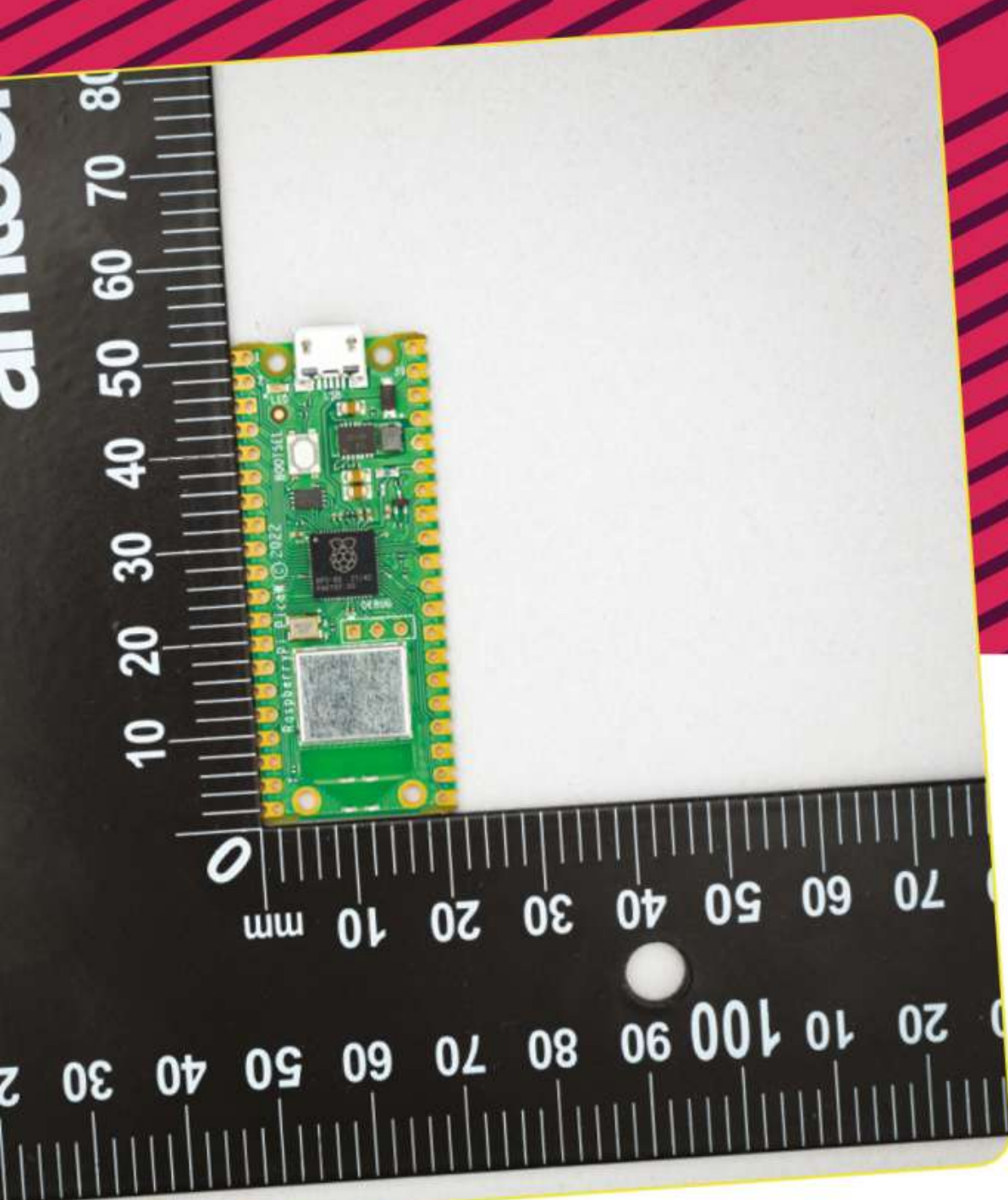
3 SQUARE GROUND PIN

Look closely at the ground pins. These have square edges instead of round ones, making them easier to locate when you are prototyping.

RASPBERRY PI PICO W GPIO GUIDE

UART0 TX - GP0 SDA - SPI0 RX - GP8	GP28 - ADC1 - GP10	GP29 - ADC2 - GP11	GP30 - ADC3 - GP12
UART0 RX - GP1 SDA - SPI0 TX - GP9	GP27 - AGND - GP7	GP26 - AGND - GP6	GP25 - AGND - GP5
GP1 SDA - SPI0 SCK - GP2	GP24 - AGND - GP4	GP23 - AGND - GP3	GP22 - AGND - GP2
GP2 SDA - SPI0 TX - GP10	GP21 - AGND - GP1	GP20 - AGND - GP0	GP19 - AGND - GP0
GP3 SDA - SPI0 RX - GP8	GP18 - AGND - GP0	GP17 - AGND - GP0	GP16 - AGND - GP0
GP4 SDA - SPI0 TX - GP9	GP15 - AGND - GP0	GP14 - AGND - GP0	GP13 - AGND - GP0
GP5 SDA - SPI0 SCK - GP2	GP12 - AGND - GP0	GP11 - AGND - GP0	GP10 - AGND - GP0
GP6 SDA - SPI0 TX - GP10	GP9 - AGND - GP0	GP8 - AGND - GP0	GP7 - AGND - GP0
GP7 SDA - SPI0 RX - GP8	GP6 - AGND - GP0	GP5 - AGND - GP0	GP4 - AGND - GP0
GP8 SDA - SPI0 TX - GP9	GP3 - AGND - GP0	GP2 - AGND - GP0	GP1 - AGND - GP0
GP9 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP10 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP11 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP12 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP13 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP14 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP15 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP16 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP17 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP18 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP19 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP20 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP21 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP22 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP23 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP24 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP25 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP26 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP27 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP28 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP29 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP30 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP31 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP32 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP33 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP34 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP35 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP36 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP37 SDA - SPI0 SCK - GP2	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP38 SDA - SPI0 TX - GP10	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP39 SDA - SPI0 RX - GP8	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0
GP40 SDA - SPI0 TX - GP9	GP0 - AGND - GP0	GP0 - AGND - GP0	GP0 - AGND - GP0

RP2040	Power
	Ground
	UART / UART (default)
	GPIO, I2C and PWM
	ADC
	SPI / SPI (default)
	I2C / I2C (default)
	System Control
	Debugging
Infineon 43439	GPIO



RASPBERRY PI PICO W SPECIFICATIONS

- RP2040 microcontroller with 2MB of flash storage
 - On-board single-band 2.4GHz wireless interfaces (802.11n)
 - WiFi 4 (802.11n), single-band (2.4 GHz)
 - 20MHz channels
 - Micro USB B port for power and data (and for reprogramming the flash)
 - 40-pin 21 mm × 51 mm 'DIP' style 1 mm thick PCB, with 0.1-inch through-hole pins also with edge castellations
 - 3-pin ARM Serial Wire Debug (SWD) port
- For more information see the Raspberry Pi Pico W datasheet (magpi.cc/docs)

RP2040 MICROCONTROLLER SPECIFICATIONS

- Dual-core Cortex M0+ at up to 133MHz
 - 264kByte multi-bank
 - High-performance SRAM
 - External quad-SPI flash with eXecute In Place (XIP) and 16kByte on-chip cache
 - high-performance full-crossbar bus fabric
 - On-board USB 1.1 (device or host)
 - 30 × multi-function General-Purpose IO (4 can be used for ADC)
 - 12-bit 500 kbps analogue-to-digital Converter (ADC)
 - Various digital peripherals
 - 2 × Programmable IO (PIO) blocks, 8 × state machines total
- For full details of the RP2040 microcontroller, please see the RP2040 datasheet (magpi.cc/2040datasheet)



PICO WH



A Pico W with a built-in header, called Pico WH, is also announced and will be available soon (this is an image of Pico H). In the meantime, it is possible to solder headers to Raspberry Pi Pico W. Header kits are available for just £1/\$1 (magpi.cc/picoheaders). Headers make it easy to connect Pico to HAT-like hardware and breadboards for circuit prototyping.

USING

RASPBERRY PI PICO W

Set up your Pico W and get onto the internet with MicroPython

Wireless support for Raspberry Pi Pico W is available using MicroPython, a coding language for microcontrollers based on Python. There will be support for C/C++ as well.

In this tutorial, we are going to connect Pico W to a Raspberry Pi using USB (you can use a non-Raspberry Pi computer with a different operating system if you prefer).

We will then install the latest version of MicroPython with wireless LAN support onto a new Pico W and use it to connect to a wireless network.

01 Update Pico W firmware

Pico W has a BOOTSEL mode that enables you to update the firmware via a USB port. Download the latest **firmware.uf2** file (magpi.cc/picowuf2) from the MicroPython documentation page.

Make sure your Raspberry Pi Pico W is not connected to a power source and hold down the white BOOTSEL button on the board. With the button held down, connect Pico W to your computer using a micro USB cable.

A drive called RPI-RP2 should appear in your computer's file system. Drag the **firmware.uf2** file to this drive. It will take a while for the file to copy across. Pico W will reboot when finished, and the RPI-RP2 drive will disappear and boot into MicroPython.

02 Connect over USB

When Raspberry Pi Pico W boots, it waits for you to tell it what to do. You can load a Python `.py` file from your computer, or interact directly with a read-evaluate-print loop (or REPL). MicroPython is equipped with a virtual USB serial port that can be

accessed via the micro USB connector. Make sure that Pico W is connected to your computer via USB, and that you have not held down the BOOTSEL button during connection.

Your computer should notice Pico W's serial port as a character device, most likely:

```
/dev/ttyACM0
```

Open a Terminal window and use this command to list your serial ports:

```
ls /dev/tty*
```

There may be a lot of tty connections, but MicroPython's USB serial will start with **/dev/ttyACM** if you are using a Linux computer. If using macOS, it will have the extension `.usbmodem` with a number appended to the end.

Top Tip

Serial Wire Debug

Another way to communicate with Pico W is via the SWD (Serial Wire Debug) pins. See Chapter 5: Flash Programming with SWD in the Getting Started with Pico book (magpi.cc/getstartedpico).

03 Install Minicom

We're going to use Minicom to access the serial port:

```
sudo apt install minicom
```

And connect to Pico W using:

```
minicom -o -D /dev/ttyACM0
```

You'll Need

- Raspberry Pi Pico W
magpi.cc/picow
- firmware.uf2
magpi.cc/picowuf2
- Raspberry Pi 4/400 (or alternative computer)
magpi.cc/raspberrypi400
- Wireless LAN network
- Thonny IDE
magpi.cc/thonny

The `-D` option and `/dev/` address are pointing Minicom to MicroPython's USB serial port, and the `-o` flag is a default 'just do it' option that stops Minicom initialising modem and lock files.

Top Tip

Minicom on Mac

If you are using an Apple Mac with a recent version of macOS, the serial will show up with a `.usbmodem` extension (followed by a number). First, install Homebrew (magpi.cc/homebrew) to install packages in macOS. Then enter:

```
brew install minicom
```

And connect to the board as below (replacing the number '1101' at the end with your number as listed in Step 2):

```
minicom -b 115200 -o -D /dev/tty.  
usbmodem1101
```

04 MicroPython prompt

Press the **ENTER** key a few times and you should see a prompt:

```
>>>
```

This is the MicroPython prompt and you can enter commands here directly to your Raspberry Pi Pico W. If you press **CTRL+D**, you will reboot your Pico W and see something like:

```
MPY: soft reboot  
MicroPython v1.18-454-g02496c997-dirty on  
2022-05-18; Raspberry Pi Pico W with RP2040  
Type "help()" for more information.
```

This is a good way to check the connection is working. The `>>>` prompt will reappear.

05 Connect to the network

We're going to connect Pico W to a local network using the network library. Enter the code from **network.py**, replacing the 'Wireless Network' and 'The Password' items with your own network and wireless LAN passcode.

network.py

> Language: **MicroPython**

DOWNLOAD
THE FULL CODE:



magpi.cc/internetpicow

```
001. import network
002. import time
003.
004. wlan = network.WLAN(network.STA_IF)
005. wlan.active(True)
006. wlan.connect('Wireless Network', 'The Password')
007.
008. while not wlan.isconnected() and wlan.status() >= 0:
009.     print("Waiting to connect:")
010.     time.sleep(1)
011.
012. print(wlan.ifconfig())
```

disconnect.py

> Language: **MicroPython**

```
001. # Connect to another wifi
002. wlan.disconnect();
003. wlan.connect('Other Network', 'The Other Password')
```

“ We're going to connect Pico W to a local network using the network library ”

Enter each line one at a time, pressing **ENTER** at the end of each one. When the code is entered, you should see the IP address your Raspberry Pi Pico W is using. For instance:

```
('10.3.15.196', '255.255.255.0',  
'10.3.15.1', '10.3.15.1')
```

You can use the code from **disconnect.py** to disconnect and connect to a different network.

It is possible to connect to HTTP (Hypertext Transfer Protocol) websites using sockets or urequests. There is an example of sockets in the documentation. You can handle redirects and there is limited JSON support. And you can even use sockets to build a simple web server and control hardware using the web server.

Example code for using Raspberry Pi Pico W with the internet is being created and shared by Raspberry Pi.

Take a look at the Pico W documentation on Raspberry Pi's website (magpi.cc/docs).

Pico W resources

Make sure you download and bookmark these resource files

- > Pico W datasheet magpi.cc/picowdatasheet
- > RP2040 datasheet magpi.cc/rp2040datasheet
- > Hardware design with RP2040 magpi.cc/hdrp2040

DOMINIC PLUNKETT AND LIAM FRASER ON RASPBERRY PI

PICO W



Dominic Plunkett



Liam Fraser

Senior principal hardware engineer **Dominic Plunkett** and software engineer **Liam Fraser** reveal all about Pico W

Adding wireless functionality to the ultra-small form-factor of Pico W was no easy task.

“There was a big question about physical space,” says Dominic Plunkett, senior principal hardware engineer at Raspberry Pi. “About whether we could

get the wireless chip and the antenna on Pico.”

A particular challenge was how to keep the GPIO pins and accommodate Infineon’s CYW43439 technology. A problem that was avoided with some clever re-routing. “We could have potentially lost the bottom two GPIO pins on each side,” he remarks. “We need four GPIO pins to control the wireless LAN.”

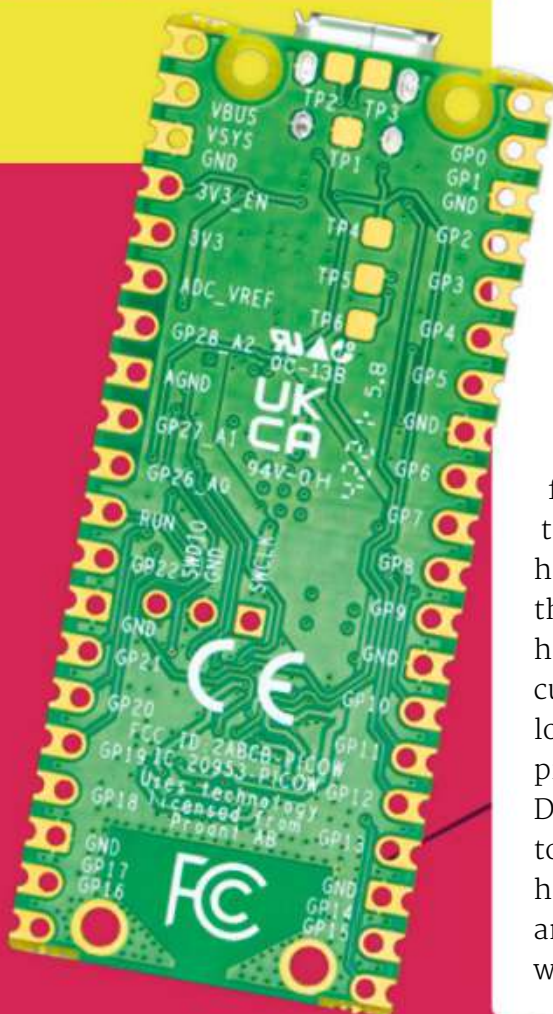
Finding a shape

Removing the GPIO pins around the antenna was tempting because it would free up space: “Antennas like space,” explains Dominic while showing us the trapezoidal-shaped feature. “And getting rid of the bottom GPIO pins would have made it easier to connect the wireless chip,” but it would have been a huge change for current users. “I didn’t want to lose any of the peripheral GPIO pins to the end-user,” says Dominic. People can add Pico W to an existing project without having to change anything and gain instant access to wireless technology.

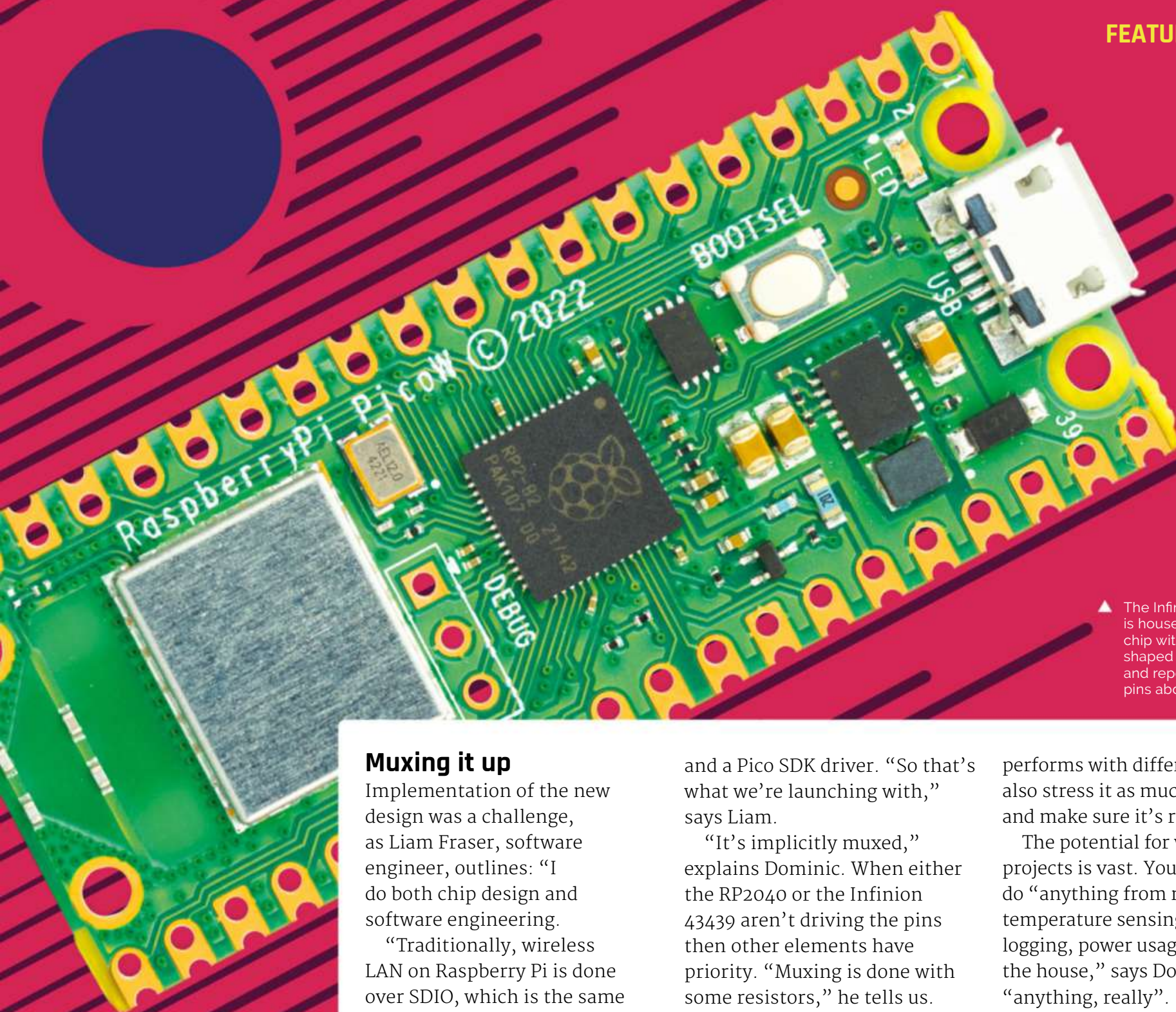
“There was a lot of work to evaluate different antenna shapes and patterns to fit in the space, and to come up with the correct balance,” says Dominic. This is one reason why the three debug pins moved from the edge of the board to a more central location: to make space for the antenna and prevent them from interfering with the wireless connection.

“After a bit of head-scratching and a long weekend of adding some secret extra resistors, we got away with using three pins to control the wireless LAN. So the three pins we’ve taken are the LED, power supply mode select, and the VBUS detect, and you get those back through the wireless LAN chip. It has some GPIOs which then go back to control the LED and power supply.

“It is a fine solution,” affirms Dominic. “A lot of people don’t use these pins. Maybe, on the first day, you might use the LED, but very quickly you go on to doing things with the I/O pins and that’s why I felt it was important to try and make sure they have the existing I/O and that it was the same.”



► Thanks to the clever design, all 40 GPIO pins remain in the same position on the newly designed Pico W board



▲ The Infineon CYW43439 is housed in an enclosure chip with the trapezoid-shaped antenna below and repositioned DEBUG pins above

Muxing it up

Implementation of the new design was a challenge, as Liam Fraser, software engineer, outlines: “I do both chip design and software engineering.

“Traditionally, wireless LAN on Raspberry Pi is done over SDIO, which is the same interface as the microSD card. However, SPI lets you do it in fewer pins and then we’ve got muxing on the pins to make all of that fit into just the three pins.

“We are amongst the first to use the SPI mode,” explains Liam. “One of the challenges was to come up with a code base that worked for MicroPython and Pico SDK,” he tells us. “The Infineon supplied code required lots of libraries to be pulled in and only worked with FreeRTOS, which is not suitable for Pico SDK or MicroPython.”

But it turned out that MicroPython already had a wireless LAN driver for a similar Infineon chip. “We were able to take that code and extend it to use this new SPI method,” reveals Liam. That provided a base for the MicroPython driver

and a Pico SDK driver. “So that’s what we’re launching with,” says Liam.

“It’s implicitly muxed,” explains Dominic. When either the RP2040 or the Infineon 43439 aren’t driving the pins then other elements have priority. “Muxing is done with some resistors,” he tells us.

“I felt it was important to try and make sure they have the existing I/O”

“There are some nice MicroPython libraries you can use, like requests. So if you want to talk to a REST API, it’s quite easy to do that in MicroPython. You just say ‘connect to my access point’ and once connected, you can send and receive data.

“The main thing we’ve been using at the moment is iPerf (iperf.fr), which is network speed testing. It’s been good for us to see how well the device


performs with different code and also stress it as much as possible and make sure it’s reliable.”

The potential for wireless projects is vast. You can do “anything from remote temperature sensing, logging, power usage around the house,” says Dominic: “anything, really”.

“You can even run small web servers on them,” adds Liam. “So if you want a basic web page where you click a button to set or get a GPIO pin, then that’s possible.”

What about Pico WH?

Pico WH is a Pico W with a plastic header attached to make prototyping and connecting to accessories easier. This is in development because it requires some further design changes. The plastic header shrouds hold the sides, and that takes up even more space. And because Pico WH has a different debug connector – vertical rather than horizontal – this will need redesigning to work with the new serial debug pins on Pico W.

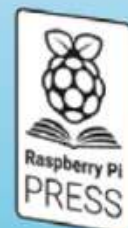
“Pico WH is on its way, but is not available yet,” Dominic tells us. 

THE OFFICIAL RASPBERRY PI PICO GUIDE

Get started with **MicroPython** on Raspberry Pi Pico

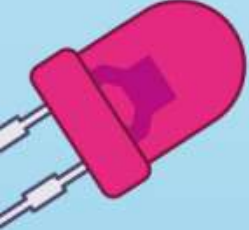


by Gareth Halfacree
and Ben Everard






Get started with MicroPython on Raspberry Pi Pico



Learn how to use your new Raspberry Pi Pico microcontroller board and program it using MicroPython. Connect hardware to make your Pico interact with the world around it. Create your own electro-mechanical projects, whether for fun or to make your life easier.



OFFICIAL
RASPBERRY PI
PICO GUIDE

- Set up your Raspberry Pi Pico and start using it
 - Start writing programs using MicroPython
 - Control and sense electronic components
 - Discover how to use Pico's unique Programmable IO
- 

Available now: magpi.cc/picobook

Build a weather station with a web dashboard



Phil King

Long-time contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

With the Pimoroni Weather HAT and Sensors Kit, you can upload your data to an easily accessible web dashboard

While it's possible to build a DIY Raspberry Pi weather station from separate components and sensors, Pimoroni's Weather HAT makes the process far simpler and easier. As well as on-board BME280 (temperature, pressure, humidity) and LTR-559 (light) sensors, the HAT features a Nuvoton microcontroller with a 12-bit ADC to read analogue signals reliably from external weather sensors connected via standard RJ11 ports. It even has a mini colour LCD screen to display readings.

The Weather Sensors Kit comprises three meteorological sensors: an anemometer to

measure wind speed, a wind vane for direction, and 'tipping bucket' rainfall gauge. Alternatively, you may already have similar sensors or be able to source them elsewhere, but you'll need a couple of RJ11 connectors to plug them into the HAT – with wind speed and direction sensors routed through one connector. Either way, let's get started.

01 Set up Raspberry Pi

If you don't already have a recent version of Raspberry Pi OS written to your microSD card, use Raspberry Pi Imager (magpi.cc/imager) to do so from another computer. While you're at it, click the cog icon in Imager to access the Advanced Options. Here you can enable SSH (useful for remote operation later), set a username and password, and configure your WiFi connection. You may also want to change the hostname to something like 'weather.local', to make it easier to identify your weather station Raspberry Pi on the network (rather than using its IP address).

02 Mount the HAT

With your Raspberry Pi powered off, mount the Weather HAT on its GPIO header, with the body of the HAT over that of Raspberry Pi. You can use any Raspberry Pi model with a 40-pin header; we chose a Raspberry Pi Zero W for our setup and

You'll Need

- Raspberry Pi
- Raspberry Pi OS
- Weather HAT + Weather Sensors Kit
magpi.cc/weatherhat



▲ Our weather station's cables go into the garage where a Weather HAT-equipped Raspberry Pi is located – an external weather-proof box would be better, however

connected to it via SSH rather than using a monitor for setup. You can add standoffs and screws through the mounting holes next to the header (and the others if using a full-size Raspberry Pi) to secure the HAT more firmly.

03 Install the software

With Raspberry Pi connected to a monitor and keyboard, open a Terminal window. Alternatively, access it remotely via SSH from another device. Enter the following commands to install the Weather HAT library:

```
git clone https://github.com/pimoroni/
weatherhat-python
cd weatherhat-python
sudo ./install.sh
```

This will automatically enable the I2C and SPI interfaces and install some additional software required for the HAT to work. In addition, the folder will include an **examples** subfolder of code examples. To be able to run these, you'll need to install some extra fonts and dependencies:

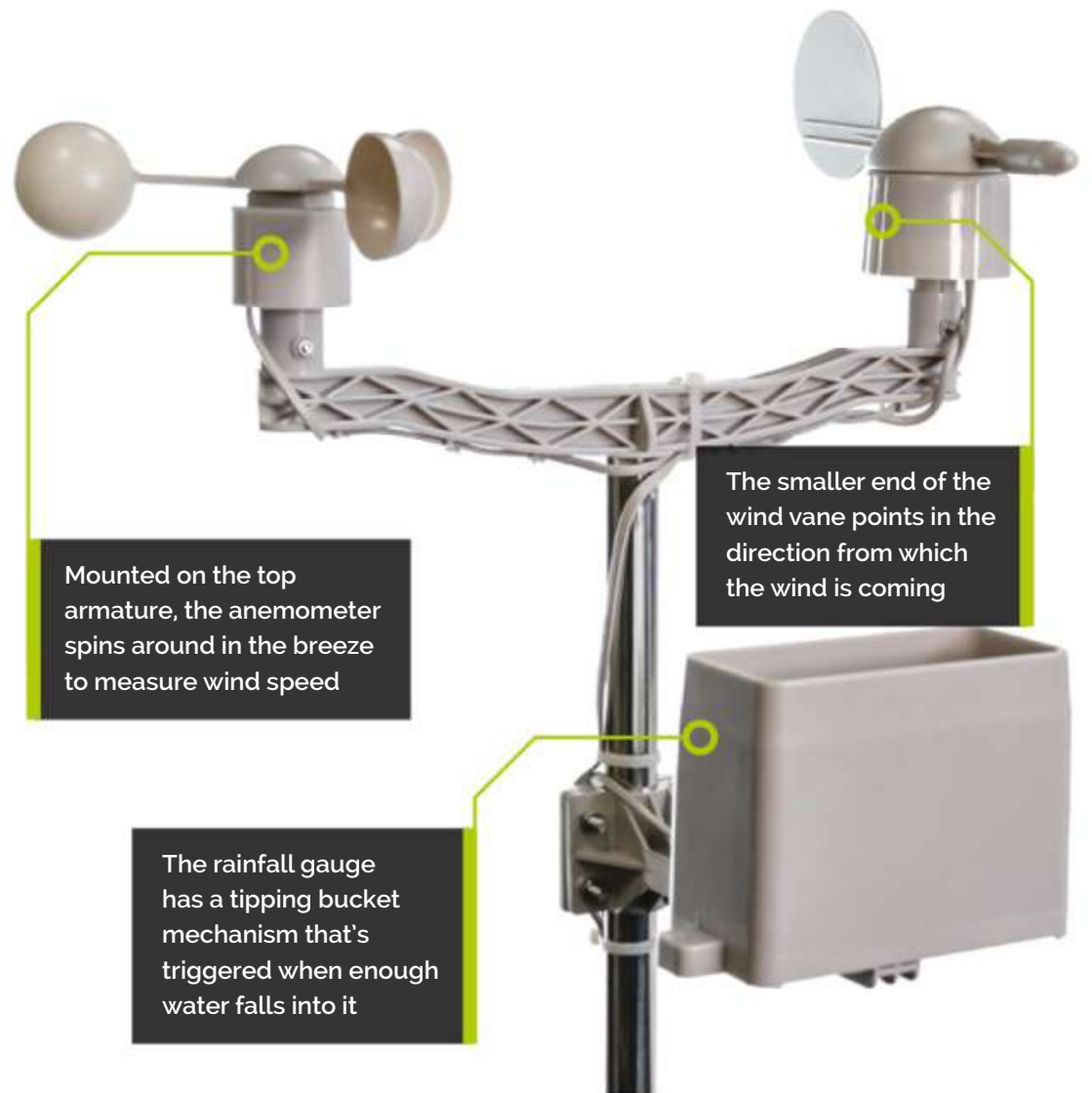
```
sudo pip3 install fonts-font-manrope pyyaml
adafruit-io numpy
```

Note: If you're using the Lite version of Raspberry Pi OS, you may also need to install some additional software before the commands above will work. To do so, enter:

```
sudo apt install python3-pip git libatlas-
base-dev
```



▲ Mounted on a Raspberry Pi Zero (hidden beneath), the Weather HAT has on-board BME280 and light sensors



Now reboot Raspberry Pi with `sudo reboot`, for the changes to take effect.

04 Initial testing

With the software installed and Raspberry Pi rebooted, let's do a quick test of the HAT's on-board sensors. Change directory to the **examples** folder and run the main Python code demo:

```
cd weatherhat-python/examples
python weather.py
```

The default screen on the LCD will show several sensor readings. Since the external sensors are yet to be connected, you will only see those for the on-board BME280 and LTR-559 sensors for now: temperature, pressure, humidity, and light. We'll test the external sensors following assembly.

05 External sensors assembly

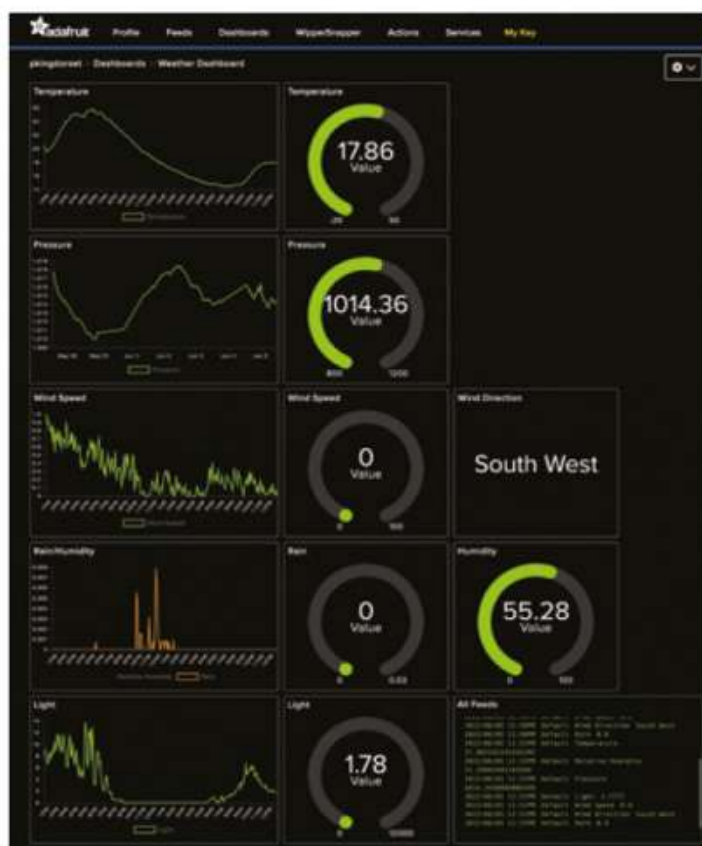
First, slide the two metal tubes together to make the mast. Then add the longer armature to the top of the mast, securing it with a screw. Mount the anemometer and wind vane securely on either side of the armature. The rain gauge fits onto a shorter armature further down the

Top Tip

Air & light

If using a weatherproof enclosure for Raspberry Pi, make sure it's well-ventilated from below. A transparent lid will also enable the light sensor to work.

► Our final Adafruit weather dashboard, using line charts, gauges, and a live stream for all feeds. You can design yours how you want it

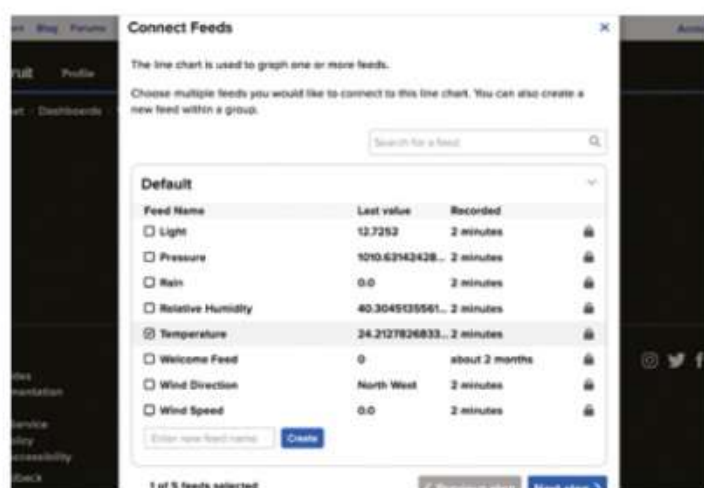


mast, clear of the other sensors so they don't interfere with its operation. For more detailed build instructions, see SparkFun's assembly guide: magpi.cc/weatherbuild.

The wires from the wind sensors can be kept neat using the built-in cable clips on the top armature. The anemometer's RJ11 connector fits into a port on the underside of the wind vane. As you'd expect, the rain gauge connector goes into the 'Rain' port on the Weather HAT, while that coming from the wind vane goes into the 'Wind' port – note that this cable is thicker, as it carries four wires (for both wind sensors) instead of two.

06 Test again

With the external sensor assembly complete, and the connectors inserted into the correct ports on the HAT, let's test they're all working correctly.



▲ Choose a block type to add to the dashboard, then select one or more sensor feeds for it

Run the **weather.py** code example, as in Step 4. This time, you should see two extra readings at the top of the LCD's default screen: for rain (in mm/s) and wind (speed in m/s). Try spinning the anemometer and you'll see the wind reading increase. Similarly, try tilting the rainfall gauge up and down to make the internal bucket tip (you'll hear a clicking noise) and you should see the rain reading rise.

Pressing the X button on the HAT will change the numerical readings to graphs. Repeatedly pressing A will show specific displays for different sensors. With the wind display selected, test the wind vane by rotating it to different positions; the graph should change to show the direction.

07 Taking it outside

There's little point getting weather readings from inside the house, so it's time to take our weather station outdoors. You could site the external sensors mast by sticking it firmly into the ground, or securing it to a downpipe with the supplied jubilee clip (gear clamp). Alternatively, as we did, you can use standard zip cable ties to secure the sensor mast to a garden fence post. You could even use a TV aerial mast to mount it on the side of a garage. Ideally, try to avoid siting the weather station too close to trees or other obstructions that may affect the wind and rain readings. Before securing it firmly in place, check Step 8 to calibrate the wind vane.

Since the sensor cables are roughly 3 m in length, you'll need to position Raspberry Pi fairly nearby unless using cable extenders. Naturally, you'll need to protect it from the rain, so you could use a standard weatherproof case or, even better, a Stevenson screen for better ventilation. We simply ran the cables through a window to Raspberry Pi in our garage, although this does give inflated temperature readings on hot days.

08 Wind vane calibration

To obtain an accurate compass direction reading from the wind vane, you'll need to make sure it's pointing north when its reading is north. To do so, use a standard compass, or an app on your smartphone, to determine in which direction north is.

The wind vane sensor has four barely visible protrusions on its body. When the shorter end of

Top Tip

Power & WiFi

You'll need a continuous power supply for Raspberry Pi, from an indoor or dedicated outdoor mains socket. You may also need a WiFi range extender if it's located far from your router.

the vane is pointing toward the protrusion nearest where the main cable (going to the HAT) comes out, it's pointing north. You can double-check this by running the **weather.py** demo code and rotating the vane so it reads north.

Now rotate the whole mast so that the north end of the wind vane is actually pointing north. Secure it tightly in place so it can't move out of position. You will also want to check that the rain gauge is level, by ensuring that the small spirit level bubble on it is centred.

09 Set up Adafruit IO

While there are numerous options for logging and charting your data locally or online, the easiest way is to use the Adafruit IO code example to set up a web dashboard that you can view from any device.

First, point a web browser to **io.adafruit.com**, click Get Started for Free, and enter your details for a free Adafruit IO account. Click on My Key to see your username and active key. On Raspberry Pi, go to the **examples** folder and edit the code file with:

```
sudo nano adafruit-io.py
```

Now alter the following lines by adding your key and username between the single quote marks:

```
ADAFRUIT_IO_KEY = 'YOUR AIO KEY HERE'
ADAFRUIT_IO_USERNAME = 'YOUR AIO USERNAME HERE'
```

Press **CTRL+X**, then **Y** to exit and save the file. Then run it with:

```
python adafruit-io.py
```

This will automatically create a Weather dashboard in your Adafruit IO account and populate it with the sensor feeds. Now, you just need to design the web dashboard.

10 Design dashboard

In your Adafruit IO account, open up the weather dashboard. Click the cog icon and Create New Block. Let's start with a temperature graph: select the Line Chart option. In the list of sensor feeds, tick Temperature and then click Next Step. Enter a block title for it, e.g. 'Temperature', leave



the other options unchanged, and click Create Block. It will now appear on the dashboard.

Let's add a real-time temperature gauge. Click the cog and Create New Block, then select the Gauge option. Tick the Temperature feed again and click Next Step, then give it a title, alter the min and max values to your preference (e.g. -20 and 50), and add 'Celsius' as the label. Click Create Block and it'll appear on the dashboard, under the chart.

Using the same process, continue to add dashboard widgets for other sensor feeds, to your preference. We also added a Stream widget to show all the live sensor feeds. You can create a chart for multiple feeds if you like, such as rain and humidity (as we did); you can then select to show/hide each feed by clicking its colour-coded rectangle in the chart key.

To reposition blocks, click the cog icon and Edit Layout, then click and drag the blocks around. You can also click the cog icon on a block to edit it, and alter the history duration for charts. When happy, click Save Layout.

▲ Use a compass or phone app to ensure the wind vane is aligned correctly to north when reading north

11 Run from boot

With your web dashboard set up and the data uploading correctly, you'll want to get your **adafruit-io.py** script to run automatically on bootup. To do so, enter **crontab -e** and add the following line at the bottom of the file:

```
@reboot python /home/pi/weatherhat-python/
examples/adafruit-io.py &
```

Press **CTRL+X**, then **Y** to save it. Now, even if the power goes off temporarily, when your Raspberry Pi reboots, the script will start running again and continue uploading data. 🌧️

Learn ARM assembly: ARM FPU



Stephen Smith

Stephen is a retired software developer who has written three books on ARM Assembly Language Programming. He is a member of the Sunshine Coast Search and Rescue and enjoys mountain biking, hiking, and running. He is also a member of the Sunshine Coast Writers and Editors Society (scwes.ca).

magpi.cc/stephensmith

Learn how to access Raspberry Pi's Floating-Point Unit (FPU) in assembly language

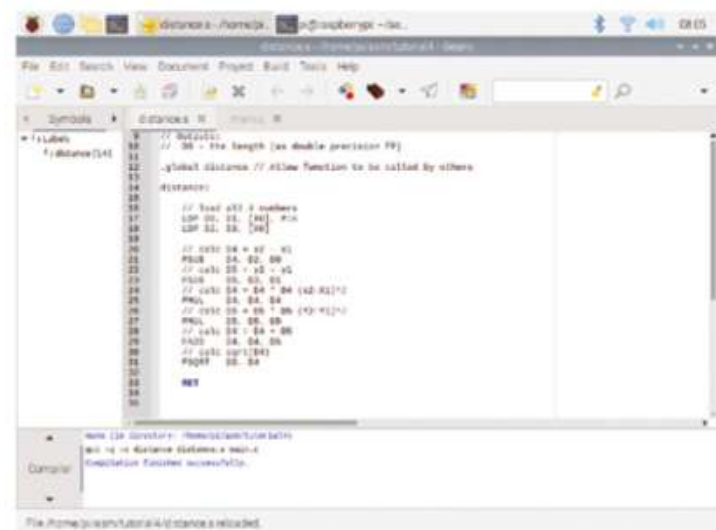
The ARM CPU used by Raspberry Pi contains both a floating-point and an ARM Advanced SIMD (NEON) vector coprocessor. In this tutorial, we'll write an assembly language function to calculate the distance between two points in two dimensions, by writing a C main program to input the points and print out the distance between them. This is typical of how larger programs are written where most of the user-interface type is written in a high-level language, but some of the low-level routines are written in assembly language for maximum performance.

This program uses ARM 64-bit assembly language, meaning that the 64-bit version of Raspberry Pi OS is required, which is available from Raspberry Pi Imager (magpi.cc/imager).

01 Create the program

The source code for this tutorial is **distance.s**, **main.c**, and **makefile**. The **makefile** contains instruction on how to compile both the C main program and the assembly language distance function, then link them together into a command-line executable program. To build and run this program, create a folder called **tutorial4** from a command prompt:

```
mkdir tutorial4
```



The source code for the assembly language distance function, and the result of running make from the build menu

Place the three source files in this folder and change directories to this folder, run **make**, and run the program:

```
cd tutorial4
make
./distance
```

You should see the output:

```
pi@raspberrypi:~/tutorial4 $ make
gcc -g -o distance distance.s main.c
pi@raspberrypi:~/tutorial4 $ ./distance
Distance between (0.0, 0.0) and (3.0, 4.0)
is 5.000000
Distance between (1.4, -2.4) and (4.6, 1.2)
is 4.816638
```


The distance routine in **distance.s** uses the formula:

```
distance = square_root((x2-x1)2 + (y2-y1)2)
```

...to compute the distance between two points (x1, y1) and (x2, y2).

02 Use floating-point registers and instructions

In 64-bit mode, the FPU and NEON coprocessors share a bank of 32 128-bit registers called V0 to V31. Only the NEON coprocessor uses all 128 bits. The FPU designates 64 bits of each register by D0 to D31 for double-precision floating-point numbers; however, these can also be used as single-precision floating-point numbers by referring to them as S0 to S31 and 16-bit half-precision floating-point designated as H0 to H31 (see **Figure 1**, overleaf).

There are floating-point instructions for addition, subtraction, multiplication, and division. There are several functions including absolute value, minimum, maximum, negation, and square root. Additionally, there are several instructions for performing conversions, performing comparisons, and moving data around. Some regular instructions, like those dealing with the stack, can take floating-point registers in addition to integer registers. Most of the floating-point instruction mnemonics start with the letter 'F', like **FADD** or **FMUL**.

03 Debug the program

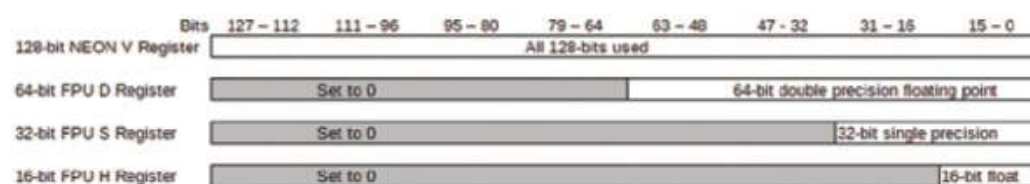
To get an idea of how the program works, we'll single-step through it in the gdb debugger.

In a Terminal window, **cd** into the **tutorial4** folder and enter the command:

```
gdb distance
```

This will start gdb and load our program. To start single-stepping through our program, first set a breakpoint at the **main** routine and then run the program to reach that point:

```
(gdb) b main
Breakpoint 1 at 0x7a0: file main.c, line 16.
(gdb) r
Starting program: /home/pi/asm/tutorial4/
distance
```



```
Breakpoint 1, main () at main.c:16
16 double point1[] = {0.0, 0.0, 3.0, 4.0};
```

▲ Format of a 128-bit coprocessor register

Next, perform three single steps through the C code to initialise the two sets of data points and execute our assembly language assembly function.

```
(gdb) s
17 double point2[] = {1.4, -2.4, 4.6, 1.2};
(gdb) s
20 dist = distance(point1);
(gdb) s
distance () at distance.s:17
17 LDP D0, D1, [X0], #16
```

The C routine passes us a 64-bit pointer to the array of four double-precision floating-point numbers which represent the two points to calculate the distance between. This one parameter is passed in register **X0**. First, the four numbers need to be loaded into floating-point registers. We accomplish this with two Load register Pair (**LDP**) instructions. The first instruction has a post index attribute that will increment the pointer by 16 bytes ready for the second instruction to load the next two values.

Now, single-step through the two **LDP** instructions:

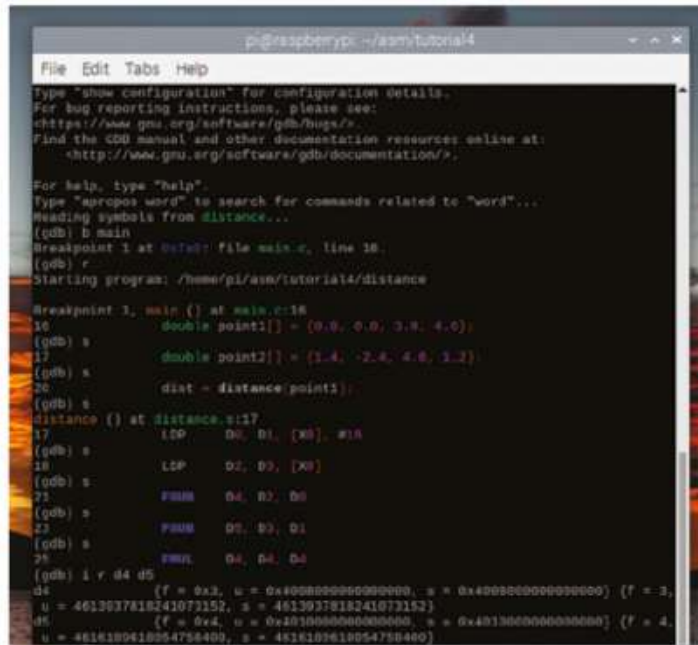
```
(gdb) s
18 LDP D2, D3, [X0]
(gdb) s
21 FSUB D4, D2, D0
```

If you enter an **info registers** command, only the standard integer registers appear. To see all the registers, enter **info all-registers**. This list is extremely long due to all the variations.

Stephen's stuff

He's written three books on Assembly Language Programming. The second one is *Programming with 64-Bit ARM Assembly Language*, which is the place to go for a deeper understanding of the topics touched on in this tutorial. The first one is *Raspberry Pi Assembly Language Programming*, and the third one is *RP2040 Assembly Language Programming* for the Raspberry Pi Pico.





► Figure 1 Single-stepping through the program in gdb

main.c

► Language: C

```
001. /*
002.  * main.c
003.  *
004.  * C main program to test the
005.  * assembly language distance function.
006.  *
007.  */
008.
009.
010. #include <stdio.h>
011.
012. extern double distance(double *point);
013.
014. int main()
015. {
016.     double point1[] = {0.0, 0.0, 3.0, 4.0};
017.     double point2[] = {1.4, -2.4, 4.6, 1.2};
018.     double dist;
019.
020.     dist = distance(point1);
021.     printf("Distance between (%.1f, %.1f) and (
022.         %.1f, %.1f) is %f\r\n",
023.         point1[0], point1[1], point1[2],
024.         point1[3], dist);
025.     dist = distance(point2);
026.     printf("Distance between (%.1f, %.1f) and (
027.         %.1f, %.1f) is %f\r\n",
028.         point2[0], point2[1], point2[2],
029.         point2[3], dist);
030.
031.     return 0;
032. }
```

However, list the registers to see the contents of interest:

```
(gdb) i r d0 d1 d2 d3
d0                {f = 0x0, u = 0x0, s = 0x0}
{f = 0, u = 0, s = 0}
d1                {f = 0x0, u = 0x0, s = 0x0}
{f = 0, u = 0, s = 0}
d2                {f = 0x3, u =
0x4008000000000000, s = 0x4008000000000000}
{f = 3, u = 4613937818241073152, s =
4613937818241073152}
d3                {f = 0x4, u =
0x4010000000000000, s = 0x4010000000000000}
{f = 4, u = 4616189618054758400, s =
4616189618054758400}
```

The second group is the decimal value and usually the most common. We see the value loaded as expected.

Now, single-step through the two FSUB instructions.

```
(gdb) s
23 FSUB D5, D3, D1
(gdb) s
25 FMUL D4, D4, D4
```

This calculates $D4 = X2 - X1$ and $D5 = Y2 - Y1$. Examine the registers to ensure the values are correct.

```
(gdb) i r d4
d4                {f = 0x3, u =
0x4008000000000000, s = 0x4008000000000000}
{f = 3, u = 4613937818241073152, s =
4613937818241073152}
(gdb) i r d5
d5                {f = 0x4, u =
0x4010000000000000, s = 0x4010000000000000}
{f = 4, u = 4616189618054758400, s =
4616189618054758400}
```

Calculate the squares by multiplying these registers by themselves. Single-step through the two FMUL instructions.

```
(gdb) s
27 FMUL D5, D5, D5
(gdb) s
29 FADD D4, D4, D5
```


Check the values of these registers to ensure the values are as expected.

Single-step through adding the two squares together with an **FADD** instruction, then calculate the square root.

```
(gdb) s
31 FSQRT D0, D4
(gdb) s
33 RET
```

We've finished the calculation. Check the result in register D0.

```
(gdb) i r d0
d0          {f = 0x5, u =
0x4014000000000000, s = 0x4014000000000000}
{f = 5, u = 4617315517961601024, s =
4617315517961601024}
```


Single-step the **RET** instruction that will return us to the C code.

```
(gdb) s
main () at main.c:21
21 printf("Distance between (%.1f, %.1f) and
(%.1f, %.1f) is %f\r\n",
```

Finish the program by using the **continue** command.

```
(gdb) cont
Continuing.
Distance between (0.0, 0.0) and (3.0, 4.0)
is 5.000000
Distance between (1.4, -2.4) and (4.6, 1.2)
is 4.816638
[Inferior 1 (process 1834) exited normally]
```

04 Next steps

Congratulations! You've learned how to call an assembly language function from C code and how to perform calculations using the ARM CPU's floating-point coprocessor. The only way to learn coding is by coding, and a good starting point is modifying and existing a working program to understand what is going on. Perhaps, code your favourite physics formula, like $E=mc^2$, or the Schwarzschild radius of a star: $R_g = 2GM/c^2$. Happy coding. 

distance.s

> Language: **Assembly Language**

```
001. //
002. // Example function to calculate the distance
003. // between two points in double precision
004. // floating point.
005. //
006. // Inputs:
007. //      X0 - pointer to the 4 FP numbers
008. //              they are x1, y1, x2, y2
009. // Outputs:
010. //      D0 - the length (as double precision FP)
011.
012. .global distance // Allow function to be called by others
013.
014. distance:
015.
016.      // load all 4 numbers
017.      LDP    D0, D1, [X0], #16
018.      LDP    D2, D3, [X0]
019.
020.      // calc D4 = x2 - x1
021.      FSUB   D4, D2, D0
022.      // calc D5 = y2 - y1
023.      FSUB   D5, D3, D1
024.      // calc D4 = D4 * D4 (x2-X1)^2
025.      FMUL   D4, D4, D4
026.      // calc D5 = D5 * D5 (Y2-Y1)^2
027.      FMUL   D5, D5, D5
028.      // calc D4 = D4 + D5
029.      FADD   D4, D4, D5
030.      // calc sqrt(D4)
031.      FSQRT  D0, D4
032.
033.      RET
034.
```

makefile

> Language: **make**

```
001.
002. all: distance
003.
004. distance: distance.s main.c
005.      gcc -g -o distance distance.s main.c
```

**DOWNLOAD
THE FULL CODE:**



magpi.cc/learnassembly4

Build a digital-to-analogue converter

Output a range of voltages using just a handful of resistors



Ben Everard

@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

Microcontrollers are great for understanding and producing digital signals. These electrical equivalents of 1s and 0s power the vast majority of our electrical devices. However, the world is not digital, so sometimes, if we're interacting with the world around us, we need our digital devices to use analogue signals. These, rather than being on or off, can range over a certain set of values.

Most microcontrollers have one or more analogue inputs which let you sample a voltage and see where in a range it falls. However, relatively few have the ability to output an analogue value. Fortunately, this is quite an easy feature to add. Let's take a look how.

If we have, say, seven digital outputs, we can use these to create an analogue output so that each digital value represents a 'bit' in final voltage.

For example, if we wanted to output the lowest possible voltage (other than 0), we'd just switch the first digital output on. If we wanted to output the highest possible voltage, we'd switch them all on. This gives us 127 possible voltages we can output (representing every combination of digital outputs turned on).

If you're familiar with binary numbers, this should all make sense. Essentially, each output is 1 bit in the binary number that corresponds to the level of the outputs. For this to work, we need each output to contribute double the voltage of the proceeding output to the final voltage total, but how can you get a digital output to contribute a certain amount to a final output?

We won't dwell upon the theory, but the circuit is a simple network of resistors (see **Figure 1**) – this is called an R-2R ladder because it can be made with two values of resistor, and it doesn't matter what they are as long as one is twice the value of the other. As you can see in **Figure 2**, we've wired it up with just a single value of resistor and placed two of them one after the other to get twice the resistance where needed. There are no hard and fast rules for what resistor values to use. Lower-value resistors will give you higher current, whereas higher-value resistors will consume less power. Bear in mind, though, that you're never going to get much current out of an R-2R ladder, so you'll need an amplifier to boost the current if you want to drive anything that requires more than a few electrons.

The electronics, then, are pretty simple. Let's look at the code.

Basically, you just need to write your output level to the GPIOs as a 7-bit binary number. Some microcontroller programming languages have the ability to set GPIOs using a mask that would let you do this in a single command. However, MicroPython does not, so we have to do it bit by bit. Fortunately, it's not too hard:

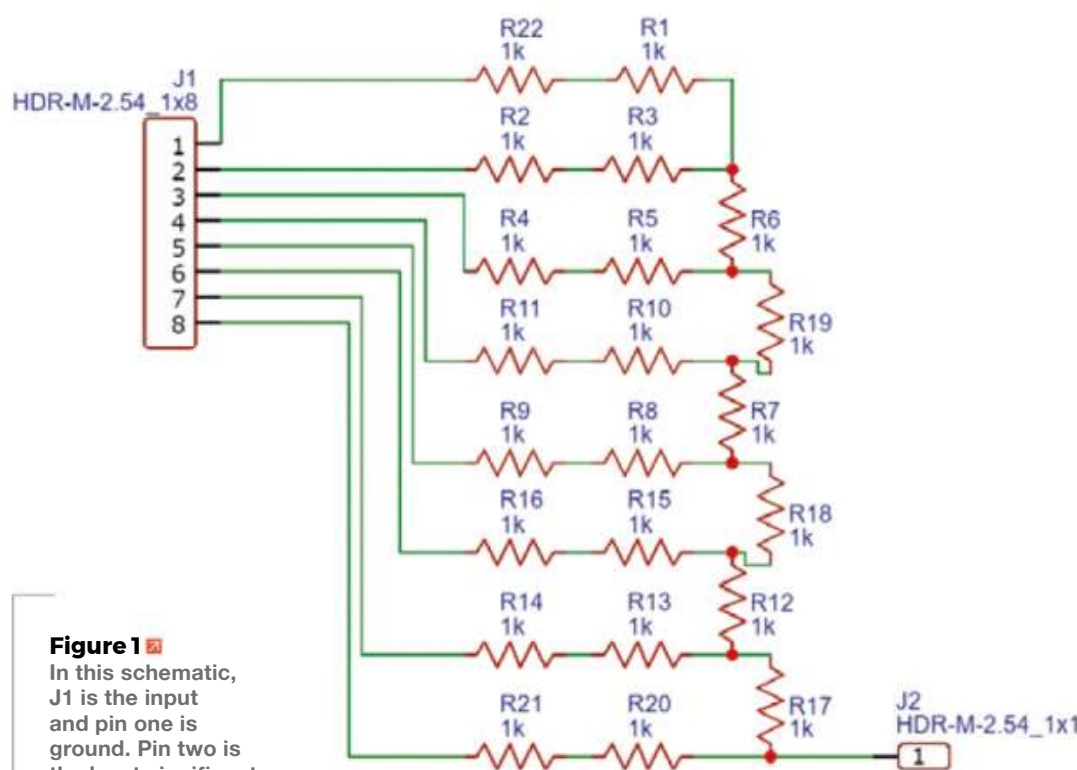


Figure 1 In this schematic, J1 is the input and pin one is ground. Pin two is the least significant bit, and it goes up from there


```
import machine
import time

pins = []
for i in range(7):
    pins.append(machine.Pin(i, machine.Pin.OUT))

def output(number, pins):
    for i in range(7):
        if 1<<i & number:
            pins[i].on()
        else:
            pins[i].off()

while True:
    for i in range(127):
        output(i, pins)
        time.sleep(0.01)
```

This code lets us set a voltage, and we can cycle through the different voltages easily enough. However, what if you want to output a particular waveform?

With low-frequency waves, you can use `time.sleep()` or `time.monotonic()` to slow down a loop to the speed you want. However, this won't be particularly accurate because Python doesn't always have predictable speeds. There's some memory management that takes place behind the scenes that can cause occasional pauses. Also, Python's not the fastest language around, so you'll be quite limited by the speed you can get out with this.

Fortunately, RP2040 has a little trick up its sleeve: Programmed input/output (PIO). This lets you attach a state machine (basically a very simple processor) to some I/O pins and do some simple processing. Crucially for us, they have very accurate timings. We've looked at PIO a few times before, so we won't cover it in detail, but the program we load into the state machine is:

```
@asm_pio(out_init=(PIO.OUT_LOW, PIO.OUT_LOW, PIO.
OUT_LOW, PIO.OUT_LOW, PIO.OUT_LOW, PIO.OUT_LOW, PIO.
OUT_LOW,), out_shift_dir=PIO.SHIFT_RIGHT
def seven_bit_dac():
    label("loop")
    pull()
    out(pins, 7) [2]
    out(pins, 7) [2]
    out(pins, 7) [2]
    out(pins, 7)
    jmp("loop")
```

We use the instruction `out(pins, 7)` to send 7 bits of data from the output shift register to the

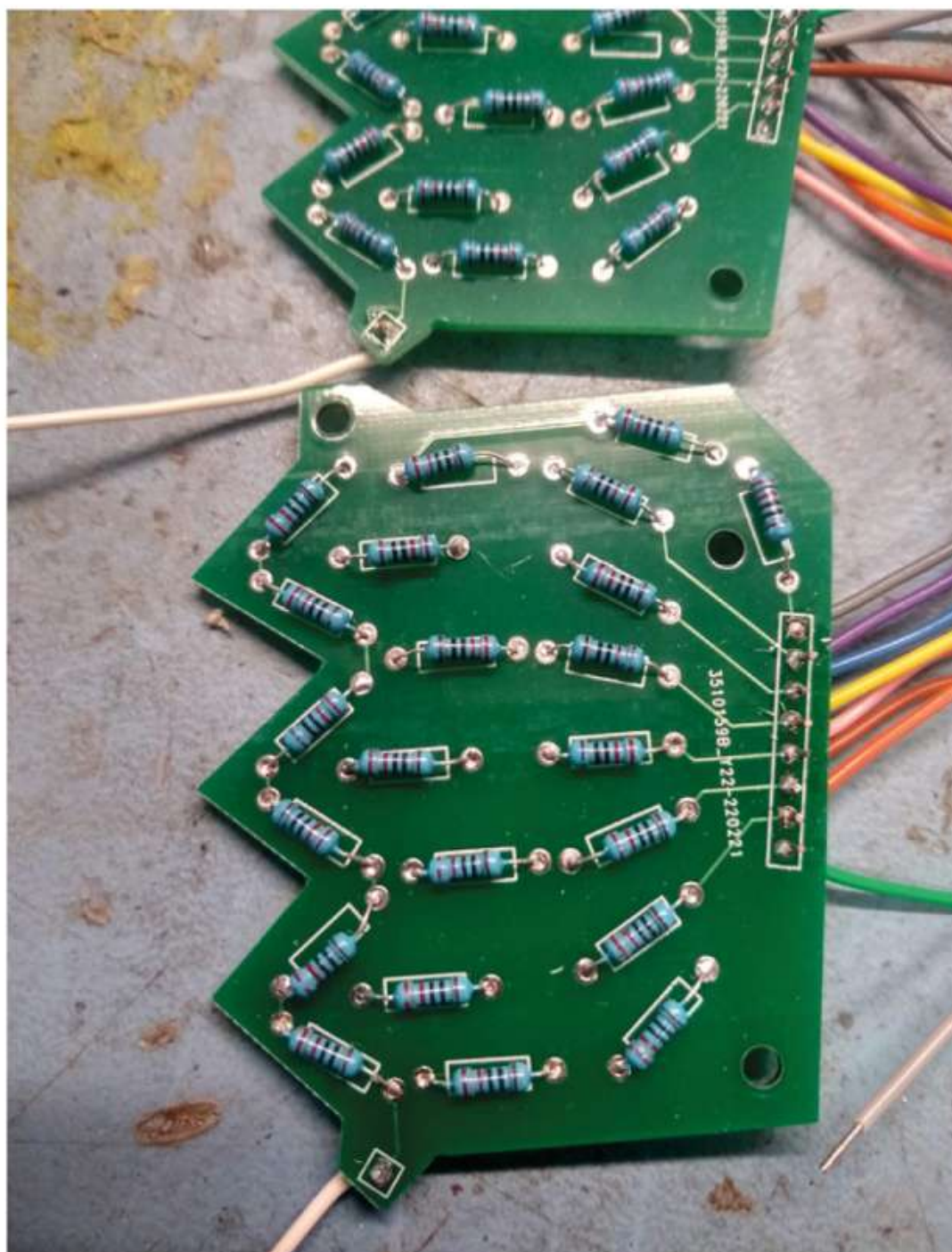


Figure 2 Just add lots and lots of the same value resistor

output pins. Each time we do this, it shuffles the data in the OSR along 7 bits. The OSR is 32 bits long, so we can only do four `out` commands before we must refill it using `pull` (which gets data from the transmit FIFO). We use the `jmp` command to turn this into an infinite loop. If you're familiar with PIO,

WHY 7 BITS?

We've built a 7-bit digital-to-analogue converter, but why this length? It's quite short and not even a standard number of bits.

Basically, it comes down to resistor accuracy. We're using a lot of resistors in this circuit, and resistors aren't perfect. In fact, they come with ratings that say what percentage of their final value they should be in. Seven bits is about the maximum accuracy you can get from 2% resistors. You can get more accurate resistors, but they're expensive, and if you need more than 7 bits, you'll probably have an easier time getting a self-contained DAC that you can control over I2C or SPI.

Right 
If you've not used MicroPython before, you can get started at hsmag.cc/WhatsMicroPython

```

Thonny - <untitled> @ 1:1
File Edit View Run Tools Help
oskill_draw_test.py <untitled> *
1 from math import floor, ceil
2 from time import sleep
3 import _thread
4
5 min_sound_freq = 110 #A2
6 max_sound_freq = 7040 #A8
7
8 @asm_pio(out_init=(PIO.OUT_HIGH, PIO.OUT_LOW, PIO.OUT_LOW, PIO.OUT_LOW, PIO.OUT_LOW, PIO.OUT_LOW, PIO.OUT_LOW, PIO.OUT_LOW), out_shift_dir=PIO.OUT_SHIFT_DEFAULT)
9 def seven_bit_dac(): # also want FIFO join.
10     wrap_target()
11     label("loop")
12     pull()
13     out(pins, 7) [2]
14     out(pins, 7) [2]
15     out(pins, 7) [2]
16     out(pins, 7)
17     jmp("loop")
18     wrap()
19
20 packed_data = array.array('I', [0 for _ in range(30)])
21 for i in range(30):
22     outval = 0
23     for j in range(4):
24         value = (i*4)+j
25         outval = outval | (value << j*7)
26     packed_data[i] = outval
27
28 sm = rp2.StateMachine(0, seven_bit_dac, freq=50000, out_base=Pin(0))
29 sm.active(1)
30
31 while True:
32     sm.put(packed_data, 0)
33
34
35
36
37
38
39
MicroPython (Raspberry Pi Pico)

```

you might notice that we could speed this up using **autopull** and **wrap**, but actually, speed isn't our main concern. We want to output audio frequencies, and there's a limit to how much we can slow down the state machine, so a bit of slowness is a good thing here.

Every single instruction in a PIO program takes exactly one clock cycle to run. You can also add optional pauses as we have done here with **[2]** – these pause for two clock cycles. We've added

these pauses so that the final instruction also has two additional instructions to run (**jump** and **pull**) before it gets back to the first **out**.

Our PIO program takes data in a slightly unusual format. We need to load the OSR with 32 bits of data which consist of four 7-bit numbers (plus four spare bits). We can create an array for this with the following:

```

packed_data = array.array('I', [0 for _ in
range(30)])
for i in range(30):
    outval = 0
    for j in range(4):
        value = (i*4)+j
        outval = outval | (value << j*7)
    packed_data[i] = outval

```

The standard Python notation for arrays are really called lists. They work well as they let you store all sorts of data and retrieve it easily. However, in our case, we need something more analogous to an array in C. That is one that has a very specific binary form. For this, we can use the array module. The array is created with a type code. 'I' in this case means 'unsigned integer' (you can see the full range of options at hsmag.cc/Arrays).

The above code creates a sawtooth wave that ramps up from 0 to 120 and then drops straight to 0. Each item in the array **packed_data** contains four 7-bit

MAKE IT PERMANENT

Resistor DACs are great when you need an analogue output and don't have many components to hand, but they can also be a more permanent solution. After all, you won't find many components cheaper than a handful of resistors.

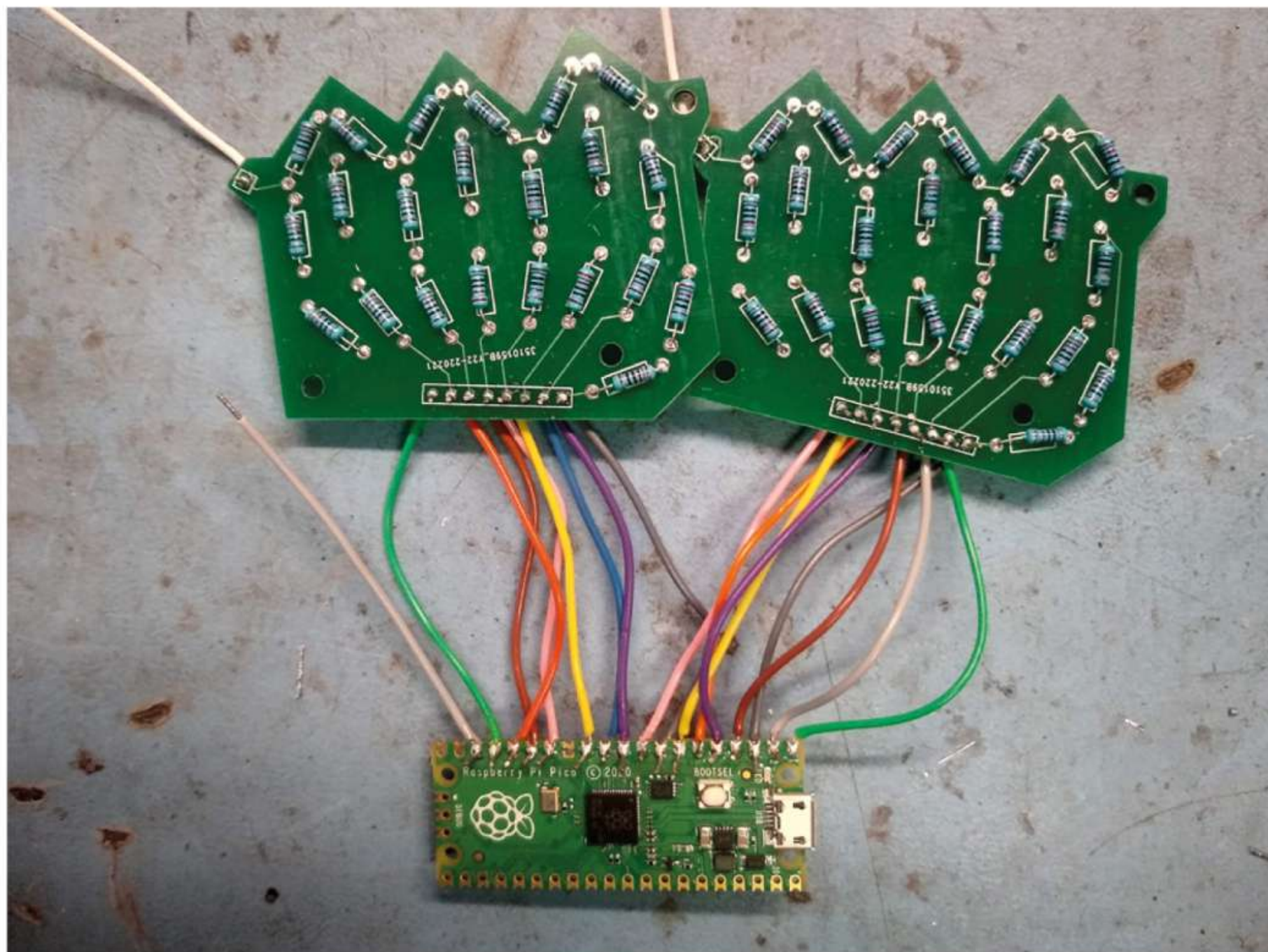
An R-2R DAC is a great project to help you learn or practice your PCB design. There's only one component (well, two if you include the headers), and the routing is quite straightforward. This author has a slightly reckless disregard for the standard that PCBs should be rectangular and components placed at right angles to each other, so the one he's created for this article (and an upcoming project) can loosely be described as punk-jellyfish.

If you need a DAC, you can design your own or use (or tweak) this author's design, which is available at: hsmag.cc/R2RDAC.



HackSpace

This tutorial is from HackSpace magazine. Each issue includes a huge variety of maker projects inside and outside of the sphere of Raspberry Pi, and also has amazing tutorials. Find out more at hsmag.cc.



values. We pack each of these four values with the two lines:

```
value = (i*4)+j
outval = outval | (value << j*7)
```

The `<<` operator shifts a value by a set number of binary digits. So, `1 << 2` would be 100 (in binary). The `|` operator is bitwise OR. This means that it ORs each bit in turn. We start with `outval` being 0, and anything OR'd with 0 is itself. This lets us set each 7-bit block of `outval` independently. We use this to place each of the four 7-bit numbers at the right place in the 32-bit binary stored in `outval`.

We now have our data and a way of outputting our data. We just need to join the two together. This is done with the main loop that simply used the state machine's `put` method to output data as fast as possible:

```
sm = rp2.StateMachine(0, seven_bit_dac,
    freq=50000, out_base=Pin(0))
sm.active(1)
```

```
while True:
```


```
    sm.put(packed_data,0)
```

The `put` method will stall if there isn't space in the FIFO for the data, so this will just feed the values into the state machine as fast as the state machine wants them.

We can set the frequency of the waveform by setting the speed of the state machine. This is done when we create it.

```
sm = rp2.StateMachine(0, seven_bit_dac,
    freq=frequency, out_base=Pin(0))
sm.active(1)
```

That's all there is to it. You can see the full code at hsmag.cc/DAC_PIO.

In this example, we're just outputting sawtooth waves at a specific frequency, but you can use this code to generate all sorts of complex waveforms at a wide range of frequencies. The state machines can run at up to 125MHz, but the maximum frequency you can output will depend on the length of your wave in data points. 

Above

You can connect more than one to a microcontroller at the same time. In this case, we've connected them to adjacent pins so that we can set both values via the same PIO program, but this time, setting 14 pins rather than 7 pins



Sean McManus

Author of *Mission Python*, *Scratch Programming in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

You'll Need

- Raspberry Pi OS
- Pimoroni HDMI 8-inch IPS LCD Screen Kit magpi.cc/8inlcd
- Picture frame (optional)
- Prepared images (see *The MagPi* 118) magpi.cc/118

➤ This IKEA Ribba picture frame is the right size for the screen, and deep enough to contain your Raspberry Pi

ArtEvolver: Build an abstract art installation

Transform Raspberry Pi into an abstract artist, with ArtEvolver. It blends images together to create surprising, stunning, and surreal artworks that constantly change

Say goodbye to boring old posters, and hello to **ArtEvolver**. Every time you look at this digital art, it's different. Fill it with patterns, paintings, and textures. Add photos of street art, structures, and neon lights. Personalise it with your children's drawings or monochrome shots of your favourite places. While you can view this on the desktop, it looks most impressive on a small screen in a picture frame. The program, based on Pygame, constantly cycles through your images and includes a safe shutdown function. To get started quickly, you can download the code at magpi.cc/artevolver.

01 Prepare your images folder

The art starts here! Store your images in a folder called **images_folder**. Ideally, the images should be resized and cropped to fit your screen size (1024×768 in my case). Last issue, we



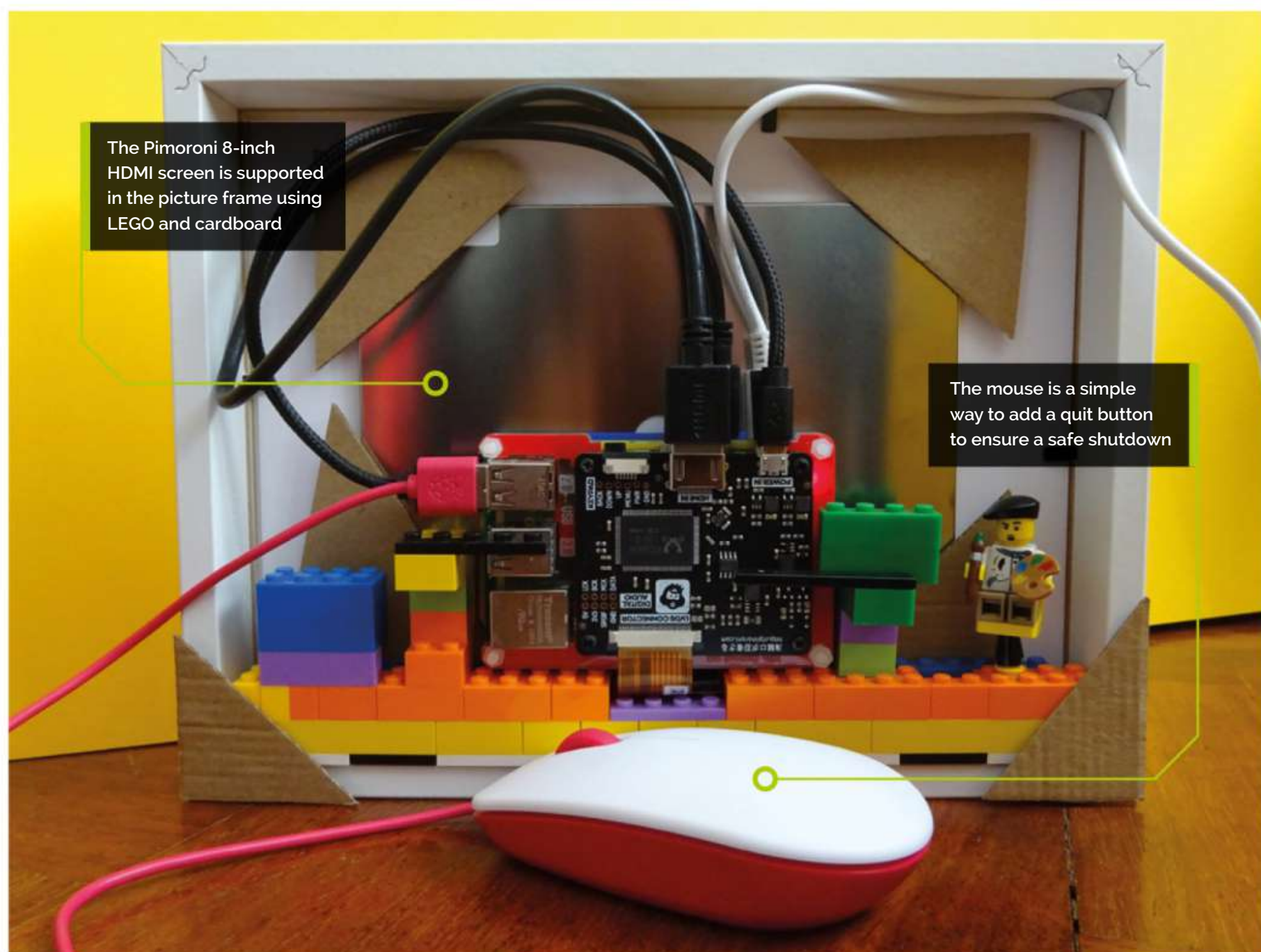
showed you how to do this with ImageMagick. It's OK to include smaller (e.g., square) or larger images, and ArtEvolver will scale them. Remove any work-in-progress images from the folder, including the **original_images** folder if you used my **landscapify.sh** script from last issue. Feel free to organise your images in subfolders.

02 Displaying the images

ArtEvolver uses Pygame to display the images. The **display_prototype.py** listing shows the simple demo created to test how the images would look, layered on top of each other. It loads three images, sets their opacity (alpha values), and then uses `windowSurface.blit()` to add them to the screen buffer. That function takes the image object and the position co-ordinates. Finally, `pygame.display.update()` refreshes the display and makes the changes visible. Change the file names in the **images** list to your own.

03 Building the pictures list

To make this work well, it needs to be able to handle hundreds of images without you needing to add them in the code. Take a look at **ArtEvolver.py**. The `index_images()` function discovers the images in the **images_folder** for you, and puts their file names (including paths) into the **pictures** list. The function is recursive. You call it with the name of the directory you want to index. When the function finds a subdirectory (using



`os.path.isdir`), it calls itself to index that directory too. This code only indexes .png and .jpg images, so rename any images that have upper-case extensions, or modify the code accordingly.

“ The alpha (opacity) value is changed to make the images fade in and out ”

04 Understanding the data structure

The `pictures` list is the primary list of all the image paths and file names. At the start of each run, it's copied to the `sequence` list, which is shuffled into a random order. Images are pulled out of the `sequence` list, and cycled through opacity values from 0 to 150, and back to 0. Each visible image is represented by a `Slide` object, containing its file name and current opacity. The `current_slide_list` stores the five `Slide` objects

that are currently on screen. When a slide fades from view, its file name is replaced in the `Slide` object with the next one from the `sequence` list.

05 Understanding opacity

Pygame is used to display the images and overlay them on each other. The `alpha` (opacity) value is changed to make the images fade in and out. We're using a maximum opacity of 150 to stop one image blocking the rest out totally. At first, we had code that either increased or decreased the opacity depending on whether a slide was fading in or out. We've simplified that by using opacity values from -150 to +150. Now, we just add 1 each time around the loop. Any negative values are made positive using Python's `abs()` function before the opacity value is used. Values go from maximum opacity (-150) to totally transparent (0) and back to peak opacity (150). At 0, the slide's image changes. At 150, the opacity value is changed to -150, and the cycle repeats.

Top Tip

Make a digital photo frame

You can turn this project into a digital photo frame. Simply change the `starting_opacities` list to `[0]` to make it display one image at a time.

► Here, two pieces of street art have been merged with ink clouds in water to create a textured, colourful image



Top Tip

Shorter is better

Search eBay for the shortest USB and HDMI cables you can find, so they fit more easily into the picture frame.

06 Setting up the slides

If all the slides start with the same opacity, you have a single composite image that fades in and out. It's more interesting to have each slide fade in and out separately, so the art is in flux. The `starting_opacities` list sets the initial values for opacity. Add or remove values in this list to change the number of simultaneous slides. We found these values by experimentation.

07 Scaling the images

Although it's a good idea to resize your image files, ArtEvolver uses the `pygame.transform.scale()` function to scale images to fit the screen. For portrait images, the scaling factor will be the image height divided by the window height. For example, if your image height was 1000 pixels and the window was 500

pixels high, the scaling factor would be 2. When the image height is set to the window height, the image width is divided by the scaling factor so it remains proportional. Landscape images are scaled to fit the window horizontally, with the height scaled proportionally. Your images don't all have to fill the window. Some of my most effective ones appear in a stripe across the middle.

08 Centring the images

Images are positioned in the middle of the window. You have to give the `windowSurface.blit()` function the co-ordinate for the top-left corner of the image. To calculate what the x co-ordinate should be, we take the middle of the window (the width divided by 2), and subtract half the image width. We do something similar for the y co-ordinate. Images that fill the window will in any case end up positioned in the top left at (0,0). However, this method neatly places images that don't fill the window.

“ We decided the simplest solution was to plug in a spare mouse ”

09 Add a quit function

We wanted to add an off button to ensure there is a safe shutdown, rather than just pulling the plug to turn it off. You could look at adding a HAT or wiring up your own button to the GPIO pins. However, we decided the simplest solution was to plug in a spare mouse. The frame is on our

display_prototype.py

► Language: Python

```
001. #ArtEvolver prototype - layers three images
002. import pygame
003. pygame.init()
004. windowSurface = pygame.display.set_mode((1024, 768))
005.
006. images = ["images_folder/image1.jpg",
             "images_folder/image2.jpg", "images_folder/image3.jpg"]
007. opacities = [45, 90, 135]
008.
009. windowSurface.fill((255,255,255))
010. for i in range(3):
011.     image_to_show = pygame.image.load(images[i])
012.     image_to_show.set_alpha(opacities[i])
013.     windowSurface.blit(image_to_show, (0, 0))
014. pygame.display.update()
```



▲ The program centres portrait-shaped images, such as this one of a man at a window

ArtEvolver.py

> Language: Python

DOWNLOAD
THE FULL CODE:

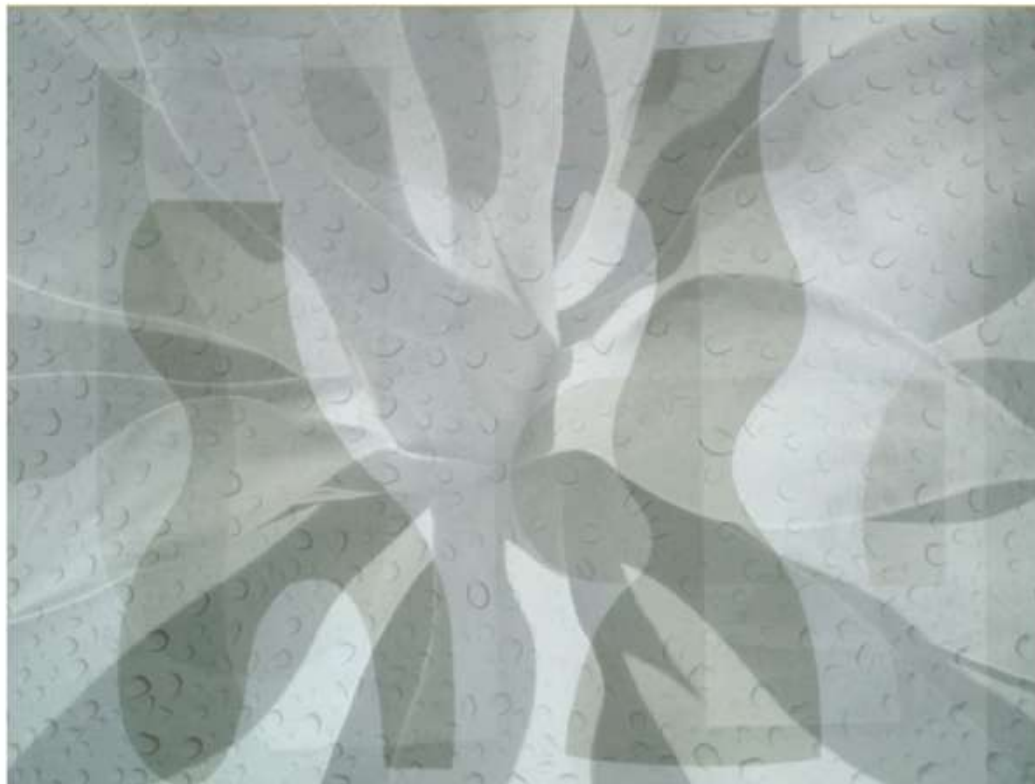


magpi.cc/artevolver

```

001. # ArtEvolver - by Sean McManus - www.sean.co.uk
002. import pygame, random, os
003. pygame.init()
004.
005. win_width = 1024
006. win_height = 768
007. windowSurface = pygame.display.set_mode((win_width,
008. win_height))
009. pygame.display.set_caption('ArtEvolver')
010. pygame.mouse.set_visible(False)
011.
012. class Slide:
013.     def __init__(self, filename, opacity):
014.         self.filename = filename
015.         self.opacity = opacity
016.
017. def index_images(path, images_list):
018.     for dir_or_file in os.listdir(path):
019.         path_plus_dir_or_file = os.path.join(
020. path, dir_or_file)
021.         if os.path.isdir(path_plus_dir_or_file):
022.             index_images(
023. path_plus_dir_or_file, images_list)
024.         elif dir_or_file.lower().endswith('.png') or
025. dir_or_file.lower().endswith('.jpg'):
026.             images_list.append(path_plus_dir_or_file)
027.     return images_list
028.
029. pictures = index_images("images_folder", [])
030.
031. while True:
032.     sequence = pictures.copy()
033.     random.shuffle(sequence)
034.
035.     # Set up initial list of current slides
036.     current_slide_list = []
037.     starting_opacities = [-90, -45, 45, 90, 135]
038.     for layer_opacity in starting_opacities:
039.         this_image = sequence.pop(0)
040.         this_slide = Slide(this_image, layer_opacity)
041.         current_slide_list.append(this_slide)
042.
043.     while len(sequence) > 0:
044.         windowSurface.fill((0,0,0)) # Black
045.         for this_slide in current_slide_list:
046.             image_to_show = this_slide.filename
047.             new_opacity = this_slide.opacity + 1
048.             if new_opacity == 150:
049.                 new_opacity = -150
050.             elif new_opacity == 0:
051.                 this_slide.filename = sequence.pop(0)
052.                 # replace image in this slide
053.                 this_slide.opacity = new_opacity
054.
055.                 image_to_show = pygame.image.load(
056. this_slide.filename)
057.                 image_width = image_to_show.get_width()
058.                 image_height = image_to_show.get_height()
059.
060.                 # Images are scaled for the long side
061.                 (fit, not fill, the window)
062.                 if image_height > image_width:
063.                     scaling_factor = image_height /
064. win_height
065.                     new_width = int(
066. image_width / scaling_factor)
067.                     image_to_show = pygame.transform.
068. scale(image_to_show, (new_width, win_height))
069.                 else:
070.                     scaling_factor = image_width /
071. win_width
072.                     new_height = int(
073. image_height / scaling_factor)
074.                     image_to_show = pygame.transform.
075. scale(image_to_show, (win_width, new_height))
076.
077.                 # Remove # on next line if your screen is
078.                 upside down
079.                 #image_to_show = pygame.transform.
080.                 flip(image_to_show, True, True)
081.
082.                 # get new height and width
083.                 image_width = image_to_show.get_width()
084.                 image_height = image_to_show.get_height()
085.
086.                 image_to_show.set_alpha(abs(new_opacity))
087.                 windowSurface.blit(image_to_show,
088.                                     (int(win_width/2) -
089. int((0.5*image_width)),
090.                                     int(win_height/2) -
091. int((0.5*image_height))))
092.
093.             pygame.display.update() # Shows composite
094.             after all slides have been blitted
095.             pygame.time.wait(30) # Adjust timings here if
096.             necessary
097.
098.         for event in pygame.event.get():
099.             if event.type == pygame.MOUSEBUTTONDOWN:
100.                 pygame.quit()

```

▲ Three monochrome images are blended here to make an abstract image, with an overlaid texture of water on glass

desk, so it's easy to hide the mouse behind it. The final lines in **ArtEvolver.py** use Pygame to check for a mouse click and quit the program if one is detected. A mouse click is the only way to close the program window.

“ A mouse click is the only way to close the program window ”

Top Tip

Image credits

Thanks to Jon Tyson, Parrish Freeman, Pawel Czerwinski, Sasha Freemind, Jen Theodore, Jr Korpa, Birmingham Museums Trust, Hermann Wittekopf, and Andrii Leonov who provided the images in the ArtEvolver composites shown here. Find these images, and many more, at unsplash.com.

10 Connect the screen

You can run ArtEvolver on any screen, or even just run it on your desktop. We're running it using a Pimoroni 8-inch HDMI IPS LCD Screen Kit. The display driver board connects to the top of the screen with a short cable, and connects to your Raspberry Pi computer using USB and HDMI cables. The display driver board doesn't need any GPIO pins.

11 Build the frame

The screen is inside an IKEA picture frame, which looks great. The display driver board is mounted above our Raspberry Pi board using standoffs, and the computer is in a Pibow case to insulate it from the screen. This arrangement keeps everything tidy and compact. To support the weight of the computer more easily, we mounted the screen upside down, and used LEGO to support the computer at the right height. There's a line in

the code that flips the images upside down so they look right. Delete the # at the start of that line if your screen is also upside down.

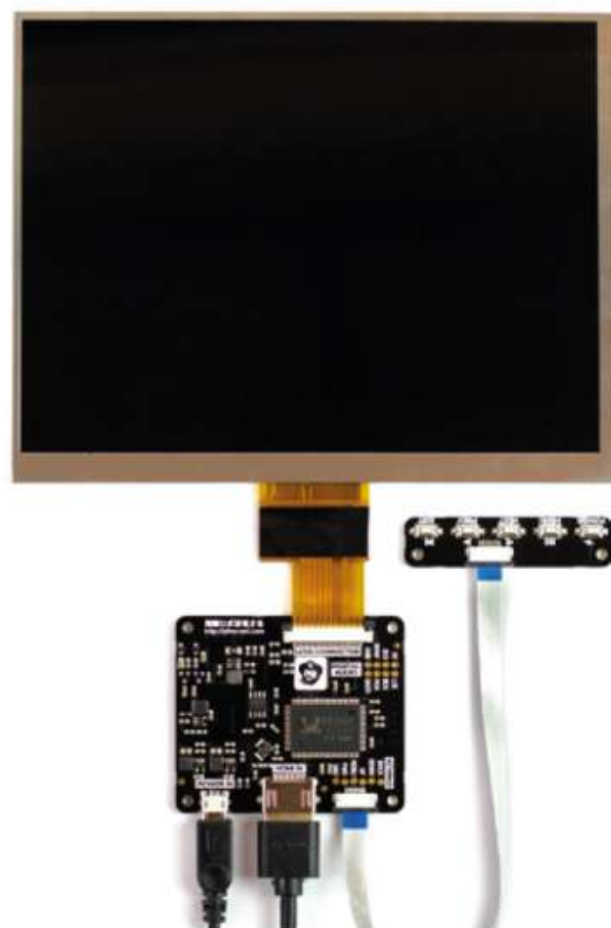
12 Make it autostart

We'll configure the **.bashrc** file to run ArtEvolver when the computer powers up and to shut down the machine safely when ArtEvolver finishes. We are using a delay of three minutes, so there's time to cancel the shutdown (using **sudo shutdown -c**) if I need to make changes, and to make sure we don't lock the machine. Note that these commands will also run if you open a Terminal window in the desktop.

Open a Terminal window and enter **sudo nano ~/.bashrc**. Add these two lines to the end of the file, save with **CTRL+O**, and exit with **CTRL+X**:

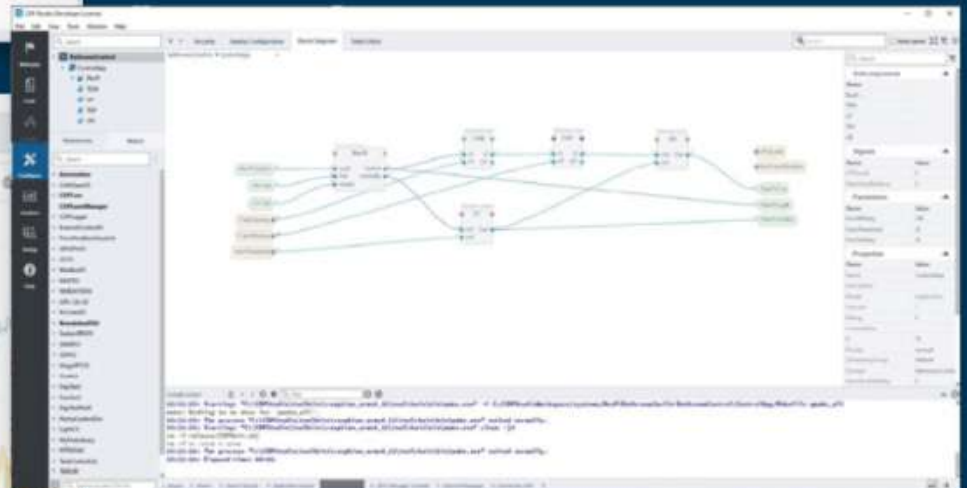
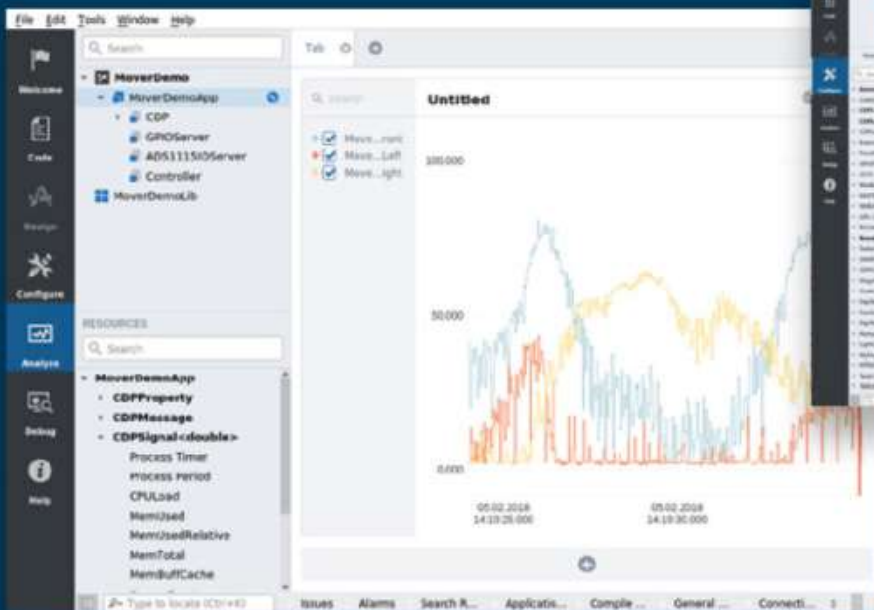
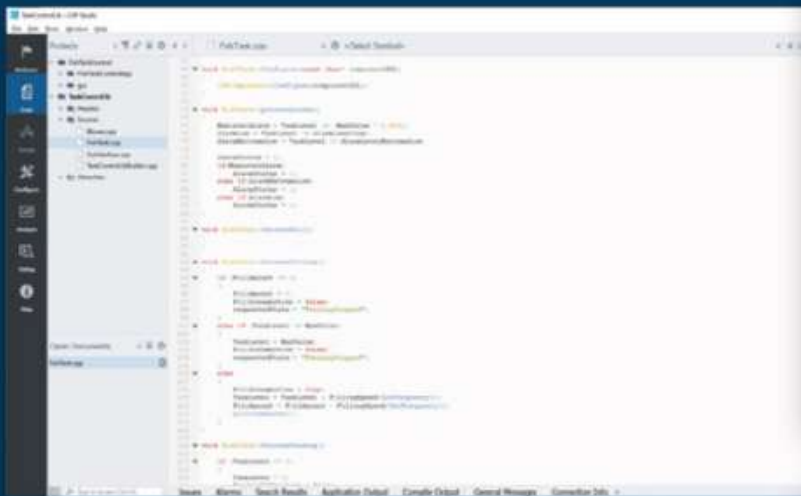
```
python ArtEvolver.py
sudo shutdown +3
```

On the desktop, go into Preferences > Raspberry Pi Configuration, and set the computer to boot to the CLI (command-line interface). (To get back to the desktop, you can enter **startx** at the command prompt). Reboot, and enjoy your digital art! 🎨



▲ The Pimoroni 8-inch HDMI IPS LCD Screen Kit includes a display driver board, which sits between the computer and the screen

Code



Configure

Analyze

With RPi 4 and
Bullseye support

PROFESSIONAL CONTROL SYSTEM DEVELOPMENT TOOL

Home projects made easy.

CDP Studio, a great software development tool for your home projects. Build systems for Raspberry Pi, use C++ or NoCode programming, open source libraries, out of the box support for GPIO, I2C, MQTT, OPC UA and more. Create beautiful user interfaces. Built for industrial control system development, **FREE for home projects.**

cdpstudio.com

Tel: +47 990 80 900 • info@cdptech.com

CDP Technologies AS // Hundsværgata 8, 6008 Ålesund, Norway





Wireframe

This tutorial first appeared in Wireframe, our sister magazine that lifts the lid on the world of video games. Every issue includes tutorials and in-depth interviews, along with news and reviews of the latest indie and triple-A games.

To find out more, visit their website at wfmag.cc.

Check out their subscription offers at wfmag.cc/subscribe.



AUTHOR
MARK VANSTONE

Code an homage to a Game & Watch classic

Revisit Nintendo's early gaming years with our homage to Fire

As the seventies turned into the eighties, Nintendo began a line of handheld games: the Game & Watch. Created by Gunpei Yokoi, these featured a liquid crystal screen and simple controls – often just left and right buttons. One such game was *Fire*, where the player had to control a pair of firefighters as they attempted to save people jumping out of a burning building. Although the moving graphics were monochrome, there was a colour overlay showing the burning building, the ground, and the ambulance.

These LCD game graphics weren't drawn with a matrix of pixels like a computer game, but had a defined set of areas on the screen where shapes could appear. This limited the variations of possible graphics in one game, and due to the speed of the electronics behind the handheld, the refresh rate was considerably slower than most computer games: only around two frames per second.

For this example, we're going to recreate the look and feel of the original *Fire* with Pygame Zero. As we have in previous editions of Source Code, we've downloaded our graphics from spriters-resource.com.

The first thing we need to do is sort out that frame rate. We want the `draw()` function to draw every update, but we only want the game objects to move every 30 updates, so what we do is have a `count` variable and only fire the `doUpdate()` function if `count%30` is zero. The `%` sign returns the remainder when we divide by 30, so we'll get numbers between 0 and 29 in this case.

We draw the background image first, and then we need to have our two firefighters move between three positions along the ground. We can catch the left and right arrow key presses, but we only want to have one movement per refresh, so we set a `movement` variable to -1 or 1 and then do the movement when the next refresh happens. We actually have three Actors for the firefighters, one for each position, and only draw the one at the current position.

Now for the people jumping out of the building. There are only 22 positions which a jumping person can be in, so we can put them in a list of tuples which represent the x and y co-ordinates of the positions on the screen. We also have a different image for each position on the screen, so as our person moves, their image is changed to

reflect the position on the screen. We start the game by making a new jumping person in the form of an actor and add that to a list of jumpers. Then, each update we move all the jumpers in the list along by one frame and change their image. When we get to the `draw()` function, we draw all our jumpers at the co-ordinates for their frame.

So now we have a game that we can control the firefighters left and right, and have people jumping out of the building following a predefined set of positions on the screen. We need to detect if the firefighters are stopping the people from hitting the ground, so we test to see if a jumper is at any of the frames where they need to be caught, and if so, are the firefighters in the right position to catch them? If not, the person will fall to the ground and the game's over. If the firefighters are in the right position to save the person all the way across the screen, however, they'll bounce into the ambulance and the player's score increases.

With that, the game's pretty much complete. The original *Fire* gave you three lives and two difficulty levels, but as always, we'll leave you to add those features in for yourself. 🕹

Python & Watch

Here's Mark's code for a *Fire*-style action game in Python. To get it running on your system, you'll first need to install Pygame Zero. You can find full instructions at wfmag.cc/pgzero.



Download
the code
from GitHub:
[wfmag.cc/
wfmag63](http://wfmag.cc/wfmag63)

```
# Fire
import pgzrun
count = catcherPos = moveCatcher = gameState = score = 0
catchers = []
jumpers = []
jumperPositions = [(130,220),(190,260),(210,320),(220,360),(240,410),
(260,360),
(270,320),(290,250),(320,220),(340,250),(360,300),
(380,360),
(390,410),(420,360),(430,300),(470,250),(500,300),
(520,360),
(538,410),(580,360),(600,320),(620,350)]
for c in range(3):
    catchers.append(Actor('catcher'+str(c), center=(240+(c*150), 425)))

def draw():
    screen.blit("background",(0,0))
    for c in range(3):
        if catcherPos == c: catchers[c].draw()
    for j in jumpers:
        if j.state == 0: j.draw()
        if j.state == -1 and count%2 == 0: j.draw()
    screen.draw.text("SAVED: "+str(score), topleft = (580, 120),
color=(0,0,0) , fontsize=25)

def update():
    global count
    count += 1
    if(count%30 == 0) : doUpdate()
    if(count%2000 == 0) : makeJumper()

def doUpdate():
    global catcherPos, moveCatcher, gameState, score
    if gameState == 0:
        catcherPos = limit(catcherPos+moveCatcher, 0, 2)
        moveCatcher = 0
        for j in jumpers:
            if (j.frame < 21 and j.state == 0):
                j.frame += 1
                j.image = "jumper"+str(j.frame)
                j.pos = jumperPositions[j.frame]
            else:
                if j.state == 0:
                    j.state = 1
                    score += 1
                makeJumper()
```

```
        if (j.frame == 4 and catcherPos != 0) or (j.frame == 12 and
catcherPos != 1) or (j.frame == 18 and catcherPos != 2):
            j.state = -1
            j.image = "jumperdropped"
            j.y += 50
            gameState = 1

def on_key_down(key):
    global moveCatcher
    if key.name == "LEFT":
        moveCatcher = -1
    if key.name == "RIGHT":
        moveCatcher = 1

def makeJumper():
    if len(jumpers)%5 == 4:
        jumpers.append(Actor('jumper0', center=(130, 270)))
        jumpers[len(jumpers)-1].frame = 1
    else:
        jumpers.append(Actor('jumper0', center=(130, 220)))
        jumpers[len(jumpers)-1].frame = 0
        jumpers[len(jumpers)-1].state = 0

def limit(n, minn, maxx):
    return max(min(maxn, n), minn)

makeJumper()

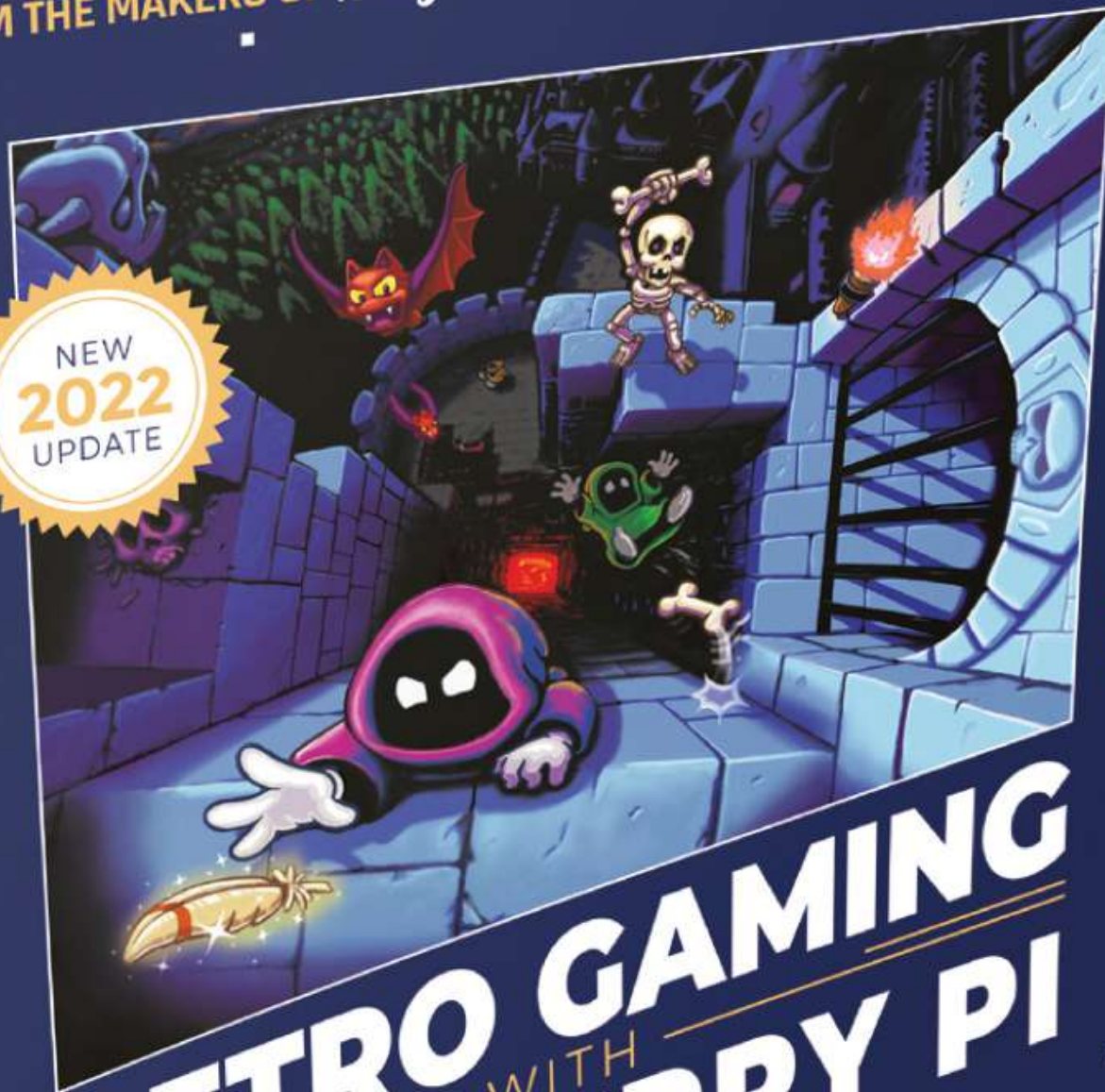
pgzrun.go()
```



^ Nintendo started releasing the Game & Watch series in 1980. *Fire* Game & Watch had two versions: the silver version and the wide-screen version.

FROM THE MAKERS OF *MagPi* THE OFFICIAL RASPBERRY PI MAGAZINE

NEW
2022
UPDATE



RETRO GAMING WITH RASPBERRY PI 2ND EDITION

164 PAGES OF
VIDEO GAME PROJECTS



**PLAY
& CODE
GAMES!**



RETRO GAMING

WITH

RASPBERRY PI

2ND EDITION

Retro Gaming with Raspberry Pi shows you how to set up a Raspberry Pi to play classic games. Build your own games console or full-size arcade cabinet, install emulation software and download classic arcade games with our step-by-step guides. Want to make games? Learn how to code your own with Python and Pygame Zero.

- ***Set up Raspberry Pi for retro gaming***
- *Emulate classic computers and consoles*
- ***Learn to code your own retro-style games***
- *Build a console, handheld, and full-size arcade machine*



BUY ONLINE: **magpi.cc/store**

REUSE YOUR RASPBERRY PI

Got an old Raspberry Pi that's feeling unloved? Join **Rob Zwetsloot** in bringing them back to life

Did you know that a lot of Raspberry Pi projects are backwards-compatible? Raspberry Pi OS still runs on original Raspberry Pi computers, and with these modern operating systems come modern builds of software and programming languages.

While a Raspberry Pi 1 may not quite have the power to run 4K video or act as a full desktop PC, there's still a lot of modern builds it can be used in. From file servers and Pi-holes, to robots and magic mirrors, it's still a powerful piece of kit.

So dig into your drawers for your older Raspberry Pi models and let's get making.

Finding a Raspberry Pi 4 – if you need a Raspberry Pi 4, keep an eye on rpilocator.com for stock alerts.

GETTING STARTED WITH AN OLDER RASPBERRY PI

Setting up a Raspberry Pi 1 is as easy as a Raspberry Pi 4

01 Installing the OS

Using the Raspberry Pi Imager tool (magpi.cc/imager) is the easiest way to install Raspberry Pi OS – or other OS – to a Raspberry Pi. You can simply install Raspberry Pi OS here; however, it's always worth looking in the Advanced settings, accessed by clicking the cog icon in the bottom right. Here you can turn on SSH for projects without a display, change the password for security, and add wireless network info, among other things. If you don't want to use a monitor and instead SSH in, make sure to use a unique name on your network.



02 Initial setup

For a first-time boot, it can be a good idea to connect a monitor, mouse, keyboard, and an Ethernet cable so you can access the internet. With a monitor, the configuration wizard will pop up and take you through some steps to set your location, change your username and password, and then download and install updates. You're done!

If you have a wireless dongle plugged in, and you didn't set the details during install, you will be able to connect here as well.

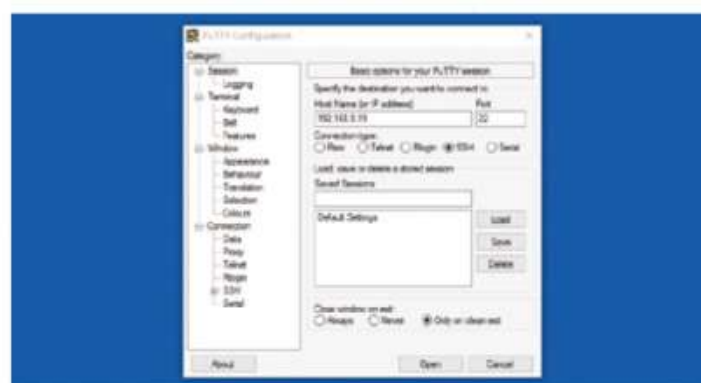


03 Headless setup

To set up a Raspberry Pi without a monitor is fairly simple, as long as you have another computer handy. We mentioned in Step 1 that you can turn SSH on during the installation to SD card phase, which you will need here. With it on, you can then use SSH to connect from your computer to Raspberry Pi over your network – in Linux and with a Mac you can do this from the UNIX terminal, and with Windows you can use PuTTY (putty.org). In PuTTY you'll just need to enter the username you set in the Imager install, followed by @, then your SSH name, e.g. **pi@raspberrypi**, and hit Open.

Once logged in, you just need to update the OS with the following two commands:

```
sudo apt update
sudo apt dist-upgrade
```



Wireless dongles
You can add wireless internet to older Raspberry Pi computers using WiFi dongles. You can find a good list of compatible ones here: magpi.cc/wifidongles

Wired Zero

Want to connect your Raspberry Pi Zero to an Ethernet cable? You can get micro-USB Ethernet adapters that do just this!

RASPBERRY PI 1 PROJECTS

Got an original Raspberry Pi? Here's what you can do with it

FILE SERVER

magpi.cc/fileserver

This very simple project just requires you to have a Raspberry Pi hooked up to a network with some added storage. It's a classic project from the early days of Raspberry Pi that involves you creating a Samba share that you can access over your network, either passworded or not, depending on your level of internal security.

Raspberry Pi 1 can easily handle this task; however, it is limited to 100MB network speeds on a wired connection – although so were all Raspberry Pi computers until Raspberry Pi 3B+.



▲ Setting up the Samba file is very important for the system

▲ You can always upgrade your server to Raspberry Pi 4 later on

Will also work on Raspberry Pi Zero!

MAGIC MIRROR

magicmirror.builders

We hate to shatter the illusion, but magic mirrors are really just big computer displays with a nice mirror coating and (sometimes) a nice frame. Raspberry Pi 1 is still small and can still deliver video via HDMI, so it's perfect for this kind of project. It's a simple display as well, so Raspberry Pi 1 can easily handle it.

We suggest adding a WiFi dongle to a project like this, if only to keep some of the wiring down. Unless, of course, you have Ethernet ports at every plug socket.



▲ The simple display is easy for Raspberry Pi 1 – which can play 1080p video just fine



▲ You don't need many components to build a smart mirror – they're just quite large

INTERNET RADIO

magpi.cc/rp1radio

With an internet connection and a speaker that can plug into a 3.5 mm audio jack, you can very quickly create an internet radio. There have been many builds over the years with varying levels of complexity, and we're a bit of a fan of analogue dials that switch between stations just like with FM radio – albeit without the fine-tuning.

You can always just have it change via buttons, via web control, or even just play the one station you prefer.



▲ You'll only need a few components for this



► This project also makes use of Arduino for a screen, although it's not necessary

HAT OR NOT

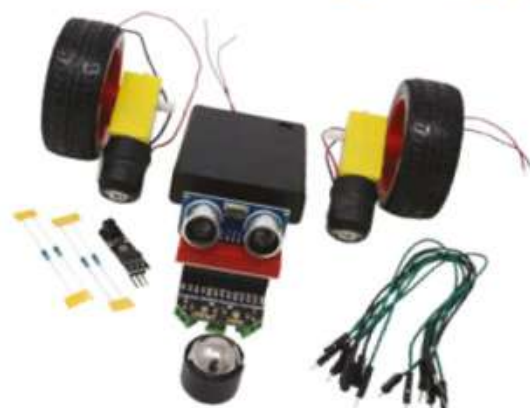
Original Raspberry Pi 1 models only have 26 GPIO pins, making them incompatible with the vast majority of HATs – they require the 40 pins of all other models. Make sure all your hardware will be compatible when choosing your project.

RC CAR

magpi.cc/rp1robot

While this is easier with a Raspberry Pi 1 B+ or Raspberry Pi Zero, due to the availability of motor controller HATs, many robots were made with original Raspberry Pi 1 computers before the B+ was released. The simplest of them allow you to control them like RC cars, much like modern robot kits.

You can also use them for robotic tasks; however, they may not perform quite as well with computer vision as a newer Raspberry Pi.



▲ The CamJam EduKit 3 is a good starter robot/RC car kit that works great with Raspberry Pi 1 B+



▲ If using an original model Raspberry Pi 1, you may need to create your own motor board

RASPBERRY PI 2 PROJECTS

More power for bigger projects



- ▲ It leans like a bike when it turns
- ▶ You can mount a Raspberry Pi Camera Module for computer vision as well



ROBOT

magpi.cc/rockyborg

With more power comes the ability for better automation and robotic tasks. Computer vision works much better on Raspberry Pi 2 and above, and with access to the full 40 pins on all models, you'll have plenty of options for robot kits.

RockyBorg is the lower-end robot kit from the robot wizards at PiBorg, which makes it great for updates and tinkering. It also looks very cool leaning into corners on its three wheels!

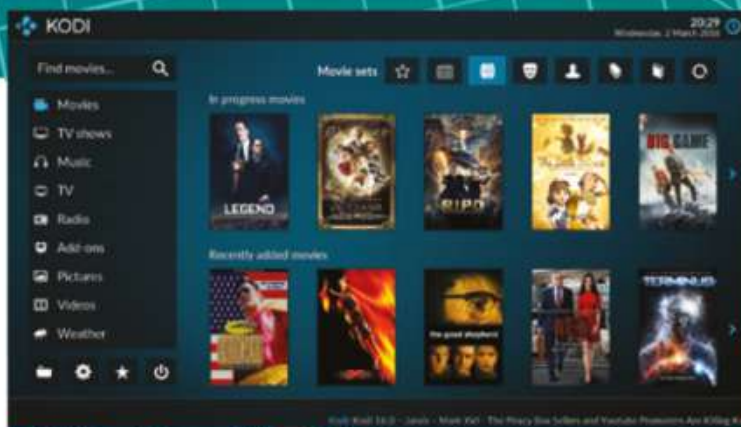
Get old and legal games for your retro gaming console here: magpi.cc/legalroms

MEDIA CENTRE

magpi.cc/102

While Raspberry Pi 1 can easily power a home theatre PC, we really feel Raspberry Pi media centres came into their own with Raspberry Pi 2. The idea is simple – load up Raspberry Pi 2 with an OS like LibreELEC that runs Kodi, have it connect to your (Raspberry Pi 1) file server, and output to your TV over HDMI.

You can get IR remotes that easily work with Kodi, or you can use an app on your phone that will connect over the network.



- ▲ Kodi has an easy-to-use interface for all your media enjoyment
- ▶ The FLIRC Raspberry Pi 3 case is perfect for Raspberry Pi 2, 3, and 3B+






▲ This project uses a Raspberry Pi Zero, but the concept is similar

Will work with
Raspberry Pi
3 and 3B+!

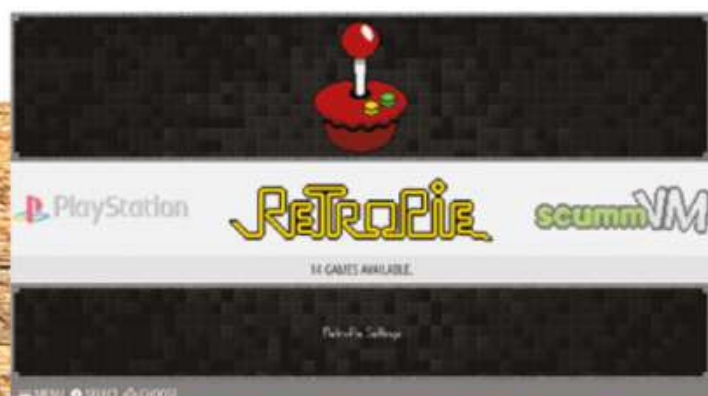
RETRO GAMING CONSOLE

retropie.org.uk

Raspberry Pi 2 can easily emulate games from the 1990s and earlier, and with RetroPie it's never been easier to set up a Raspberry Pi to play them. It takes a very short amount of time and has great configuration options for controllers of all kinds.

You can even install Kodi to it and have a hybrid entertainment PC that can play games and let you watch your videos and listen to your music as well. There's even Steam Link capabilities, so you can stream games from a gaming PC to your TV. 

- ▶ SEGA has released official ROMs via the Mega Drive collection on Steam
- ▼ RetroPie's menus are easy to navigate, and there's loads of documentation on the more advanced options



RECIPE COMPUTER

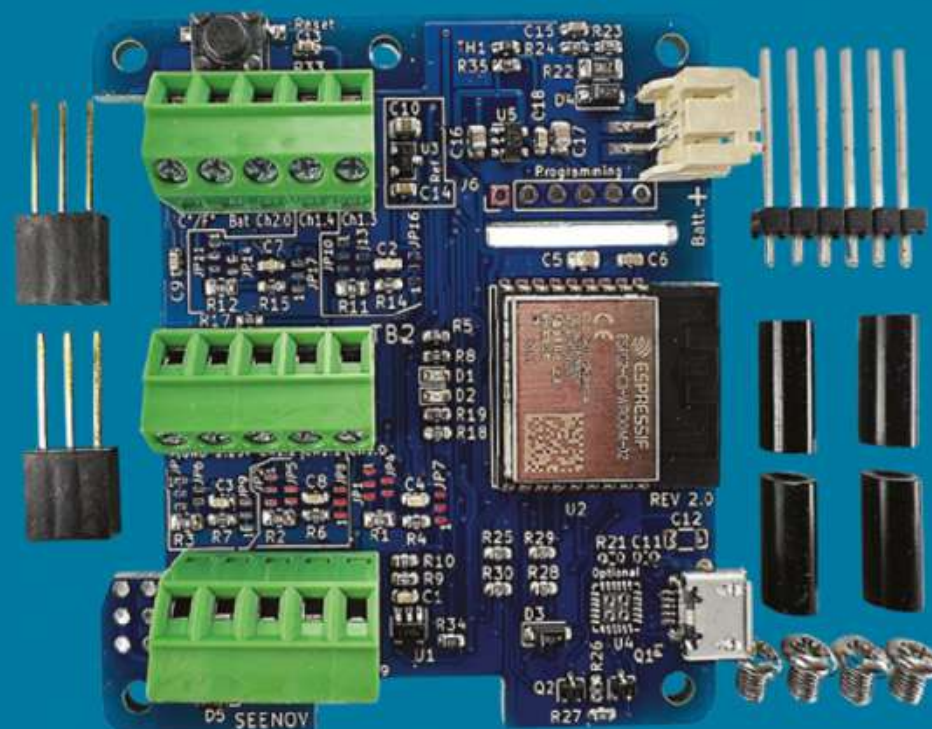
magpi.cc/kitchenpc

We've tried using Raspberry Pi 2 as a desktop computer and while it works fine, it's not a proper replacement like Raspberry Pi 4 is. However, it is very good for just browsing recipe websites or the web in general, making it perfect for mounting in your kitchen as a recipe computer.

It can even run YouTube in case you need to follow a Binging with Babish or Ann Reardon video while in the kitchen. You don't even need a fancy monitor for the whole setup either.

SEENOV®

See Innovation - Voir l'innovation



Seenov 6 channel ESP32C3
ADC HAT for the Raspberry Pi

available at
amazon

www.seenov.com



Hiwonder SpiderPi

SPECS

DIMENSIONS:

700 × 610 × 160 mm; 2.3 kg

BATTERY:

2500 mAh 11.1 V lithium-ion battery pack, 40 mins battery life

ROBOT FRAME:

18 × LX-824 three-port bus servo, 2 × LFD-01 servo, 20 × servo wire, 480 p camera, metal frame, voltage display

► Hiwonder ► magpi.cc/spiderpi ► From £480/\$600

A Raspberry Pi 4-powered hexapod can be both imposing and surprisingly agile, discovers **Rosie Hattersley**

The first thing to note about SpiderPi is it's pretty large: with its legs fully extended, this robot's body is nearly 30 cm off the ground, and spans nearly 60 cm. Even sitting next to your reviewer on the sofa while plugged in and charging, SpiderPi has a certain presence. No wonder our cat took one look and spent the rest of the day outdoors for fear of a chance encounter with this intimidating-looking hexapod.

The hardware from which SpiderPi is constructed is similar to that of its Hiwonder sibling, TonyPi, reviewed back in *The MagPi* 111 (magpi.cc/111). Controlled by a 4GB Raspberry Pi 4, SpiderPi's legs are made from 18 servos – three for each of its six legs – fitted into hinged aluminium frames, giving

it a Meccano-like appearance. This also gives you some idea of how robust and well-constructed this clever hexapod is. The robot's HD camera head can rotate through 140 degrees while observing its immediate surroundings, picking out faces and signalling recognition by waving a leg.

It's powered via an 11.1 V lithium-ion battery that connects to its governing Raspberry Pi 4, and provides roughly 40 minutes of usage from battery power. To control SpiderPi, install the WonderPi app on your tablet or smartphone, then select either a direct connection mode and use SpiderPi as a personal hotspot, or connect over a LAN or home network. Initiate a search for SpiderPi if the app doesn't automatically detect it. However, you'll realise the robot has been 'found' as its limbs will suddenly move, ready for action. Make sure you've got plenty of space around you, as you're about to discover it can cover half a room with alacrity.

“ Our cat took one look and spent the rest of the day outdoors ”

Sudden movements


Select SpiderPi when its icon appears in the app to bring up the menu of 'games', including the AI-based line tracker and coloured ball recognition activities.



► SpiderPi can scuttle in any direction surprisingly quickly



An on-screen D-pad prompts SpiderPi to scuttle forward, back, and sideways, as well as twist menacingly on its haunches and raise its front legs to fight. It can even perform an inelegant but speedy flip and boasts an 'Inverse Kinematic Gait'. SpiderPi's rate of movement and camera sweep angles for the object and face recognition games are easily adjusted via a sliding bar. A friendly wave of a leg usually indicates SpiderPi has spotted you if you're sitting within a metre of its camera, but you might be overlooked if you're at the extremes of range of vision.

This intelligent hexapod can also be tasked with following an undulating line on the floor, distinguishing between different-coloured objects and following commands issued via a Python script tied to a QR code its 480p camera notes and Raspberry Pi decodes. It can be challenged both to pick up and carry and to avoid objects, effectively shuffling round to outfox barriers and obstacles, as long as they're hefty enough for its camera to notice them. These games, of course, are ideal if you want to use SpiderPi as an interactive learning tool, since new commands can be written in Python, and QR codes generated for its camera to find and carry out. 



▲ Favourite pets may regret showing quite so much interest when SpiderPi springs into action

Verdict

SpiderPi is a solidly built robot that will impress friends and intimidate battlebot challengers. Its realistic scuttling movements, accurate face and object recognition, and the ability to learn new tricks make it a great, if pricey, learning tool.

9/10

Maker Nano RP2040

SPECS

PROCESSOR:

RP2040 with 264kB SRAM and 2MB flash storage

FEATURES:

14 × status LEDs, 2 × RGB LEDs, 3 × push-buttons, piezo buzzer, with switch

CONNECTIONS:

22 × digital GPIO pins, 4 × analogue inputs, 2 × Qwiic / STEMMA QT ports, 2 × Grove adapters

DIMENSIONS:

49.6 × 21.1 × 15 mm

► The Pi Hut ► magpi.cc/makernano ► £8 / \$8

All the power of the RP2040 chip in an Arduino Nano form factor. By **Phil King**

The Maker Nano RP2040 aims to combine the form factor of an Arduino Nano with the impressive power of the RP2040 system-on-chip, as used by Raspberry Pi Pico.

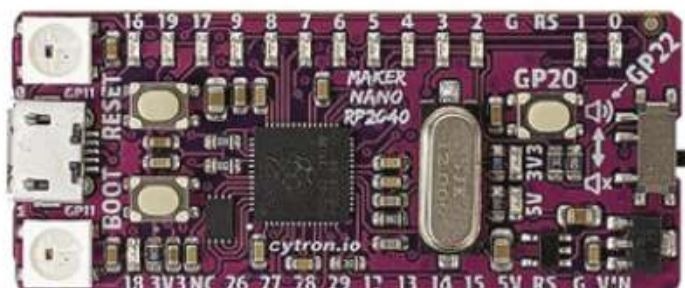
Slightly shorter than a Pico, this board has 30 pre-soldered pins, all helpfully labelled on the silkscreen. While the pinout is very similar to the Arduino Nano's, there are four (12-bit) analogue inputs rather than eight (10-bit). Some digital GPIO pins are numbered differently too.

A pair of reset pins are linked to a handy reset button on the top, where there's also a boot button – held to attach the board as a USB drive to a computer – and a programmable button linked to GPIO pin 20.

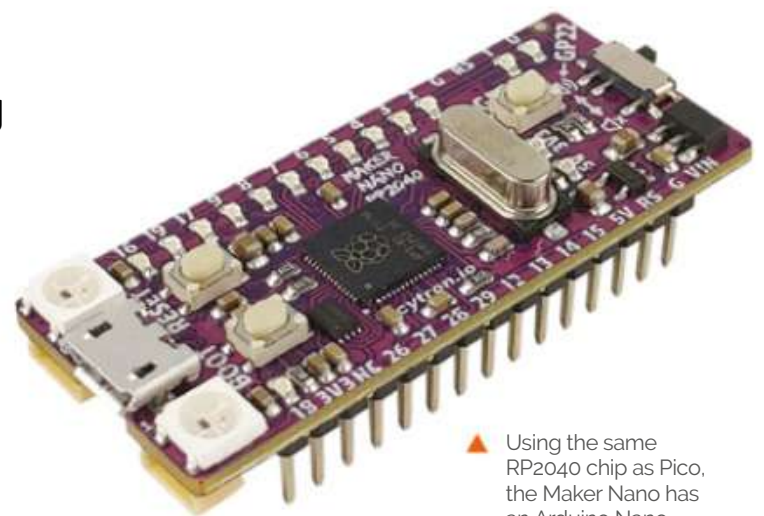
Other connections include a micro-USB port (for power and connecting a computer) and two Qwiic / STEMMA QT 'maker ports' which also work with Grove modules using the two supplied adapter cables.

Lights and sound

On either side of the micro-USB port are two RGB NeoPixel LEDs. Since both are linked to GPIO 11, they can't be controlled separately, but some



▲ Two RGB NeoPixel LEDs are found on either side of the micro-USB port, while 14 of the GPIO pins have tiny status LEDs



▲ Using the same RP2040 chip as Pico, the Maker Nano has an Arduino Nano-style pinout

cool colour-changing effects can be achieved. In addition, 14 of the digital pins have tiny blue status LEDs that may well prove useful when troubleshooting I/O connections for circuits.

“ The Maker Nano RP2040 comes with CircuitPython firmware pre-installed, including a mini demo ”

Primitive audio is also provided in the form of a piezo buzzer that can be muted with a slide switch. It can be programmed to play beepy tunes by sending numerical note values.

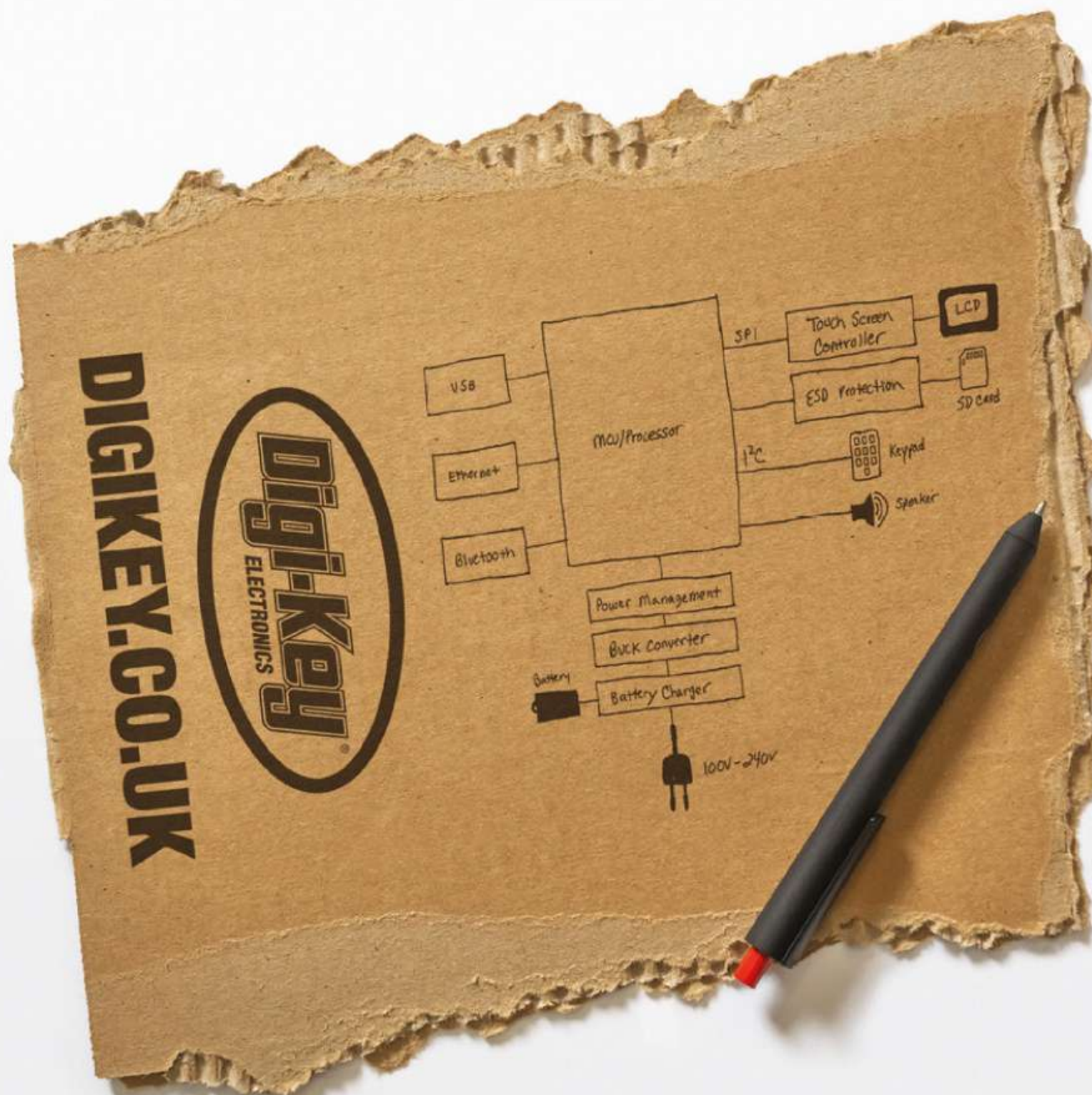
Talking of coding, the Maker Nano RP2040 comes with CircuitPython firmware pre-installed, including a mini demo that shows off the lights and sound. More code examples are found in the GitHub repo: magpi.cc/makernanogh. Alternatively, you can flash MicroPython firmware or program in C/C++ (natively or via the Arduino IDE). Like Pico, it's very flexible. ”

Verdict

A well-designed microcontroller board with a familiar Arduino form factor, the power of RP2040, and cool bonus features.

8/10

IDEAS START HERE



From millions of in-stock parts to the latest new product inventory, we've got everything you need to turn breakthrough ideas into reality.

Get inspired at [digikey.co.uk](https://www.digikey.co.uk) or call 0800 587 0991.



Digi-Key is a franchised distributor for all supplier partners. New products added daily. Digi-Key and Digi-Key Electronics are registered trademarks of Digi-Key Electronics in the U.S. and other countries. © 2022 Digi-Key Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA

ECIA MEMBER
Supporting The Authorized Channel

Pico Unicorn Pack

SPECS

FEATURES:

16×7 RGB LED matrix, 4 × programmable push-buttons

DIMENSIONS:

65 × 25 × 10 mm

► Pimoroni ► magpi.cc/picunicorn ► £22 / \$23

Light up your Pico projects with this vibrant LED matrix. By **Phil King**

Want to add some sparkle to your Pico project? The Pico Unicorn Pack offers a 16×7 matrix of bright RGB LEDs – that’s 112 in total – along with four programmable push-buttons. All you need is a Raspberry Pi Pico with soldered male pins and you can plug it into the Pico Unicorn’s dual female headers, found on the rear of the board.

You’ll also need to flash Pico with Pimoroni’s custom MicroPython UF2 firmware. This includes

“ The LEDs are updated in the background with very little CPU usage ”

the MicroPython module for the Pico Unicorn, which you can import at the top of your programs. While not as sophisticated as the Python one for Pimoroni’s Unicorn HATs for Raspberry Pi computers, it does enable you to set individual pixels to RGB values (or a single value) and read button presses. Disappointingly, there’s no built-in function for scrolling text, although it can be done with a bit of know-how.


▼ The RGB LEDs are bright and super-responsive, making use of Pico’s PIO



▲ The same size as a Pico, the Unicorn has an RGB LED matrix and four push-buttons

Pico Pong

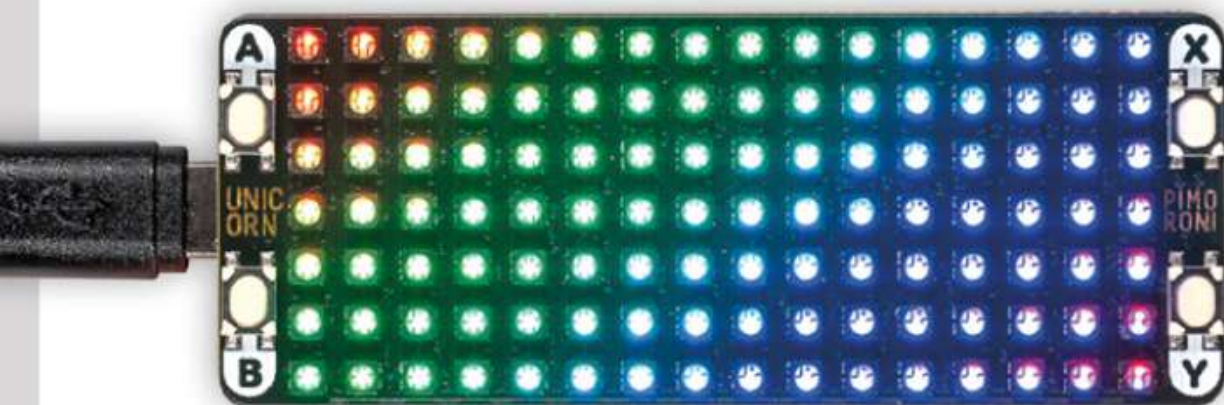
Only two MicroPython code examples for Pico Unicorn are provided in Pimoroni’s GitHub repo, but doing a search for ‘Pico Unicorn’ on GitHub reveals a wide variety of programs created by the community, giving you an idea of the possibilities. Our favourite is a two-player game of Pong using the buttons as controls (magpi.cc/unicornpong). There’s even a funky plasma generator that enables you to scroll messages using a frame buffer (magpi.cc/unicornplasma). We also found a version of Conway’s Game of Life on the Pimoroni forums.

Those RGB LEDs are bright and very responsive. Controlled using the PIO state machines on Pico’s RP2040 chip, they’re updated in the background with very little CPU usage. In fact, it’s so fast that 14 bits of resolution can be achieved, resulting in smoother brightness transitions using gamma-corrected values. In short, it looks very impressive. 

Verdict

With a matrix of super-responsive RGB LEDs, it’s ideal for animations and even games using the tiny push-buttons.

9/10



English not your mother tongue?

The MagPi is also available in German!




Subscribe to the German edition of The MagPi and get a Raspberry Pi Pico with headers and a cool welcome box **FOR FREE!**

Use the coupon code **115PicoDE**
on www.magpi.de/115



10 Amazing: Summer projects

Get outside with Raspberry Pi
and Raspberry Pi Pico

The weather's getting warmer, and the sun is appearing much more these days. This means it's time to get outside. While the WiFi outdoors might not be as good as the weather, you can still make some really cool things with your Raspberry Pi and Raspberry Pi Pico. Here's just ten... 



▲ Sailing Pi

GPS boating

From issue 59, Bill Ballard showed us how he can track himself sailing around with some custom Raspberry Pi kit for better analysis.

magpi.cc/59



▲ FarmBot

Automated farming

This Raspberry Pi-powered raised farm bed allows you to expertly plan, and then efficiently grow, your veg and herbs. Perfect for the BBQ.

farm.bot

► Photography

HQ snapping

Use Raspberry Pi HQ Camera and a snazzy case to go around snapping photos with your own pretty fancy camera setup, with interchangeable lenses as well!

magpi.cc/hqcamcase



▲ MudPi

Grow plants easily

This modular system allows you to perfectly irrigate your garden, or indoor hydroponics, and will make sure you never waste water.

mudpi.app

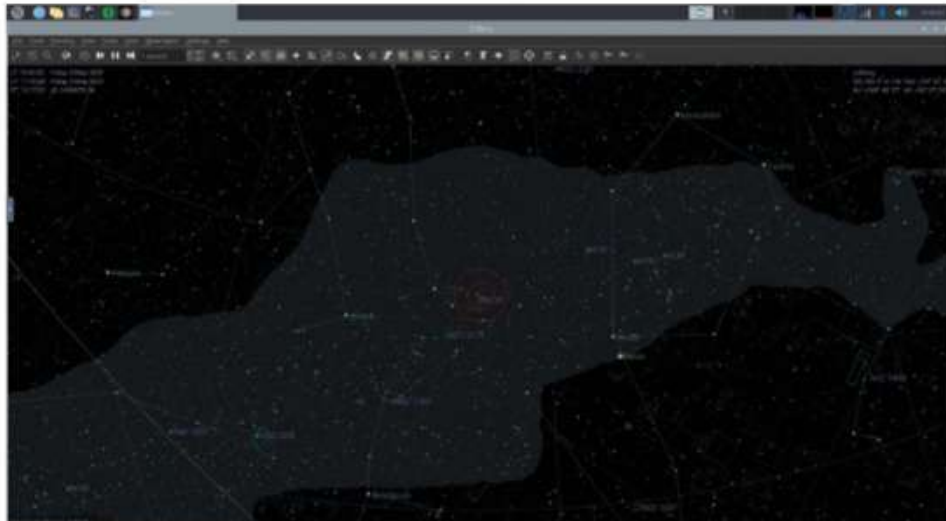


▲ MonsterBorg

RC and automated

This incredible robot platform makes for the perfect remote control beast, or even an automated racer in your own robot league.

magpi.cc/monsterborg



▲ Astroberry

Watch the skies

Create your own automated astro computer and connect it to your telescope to take your stargazing to the next level.

astroberry.io



◀ Weather station

Check the weather

A great little science experiment to do over the summer, and also a good way to make sure if you need an umbrella or not.

magpi.cc/weatherstation



▲ Hiking Companion

On-the-go

This little companion uses a Sense HAT and a GPS module to track and display several data bits, making use of the joystick on Sense HAT to move between modes.

magpi.cc/hiking



◀ Overkill Cyclometer

Smart cycling

This uses a Raspberry Pi Pico and a magnetic reed switch to track the speed of the bike, along with other analysis of environmental factors.

magpi.cc/cyclometer

► Enviro+

Sense more

If you live in a city or other kind of big built-up area, it's a good idea to be able to measure air quality, as well as the weather.

magpi.cc/enviro



Smart gardening

Check out issue 117 of *The MagPi* for more stuff you can do outside!
magpi.cc/117



Learn databasing with Raspberry Pi

Unlock the power of databases with these resources. By **Phil King**

Introduction to Databases and SQL

AUTHOR

FutureLearn / Raspberry Pi

Price:
Free


magpi.cc/flatabases

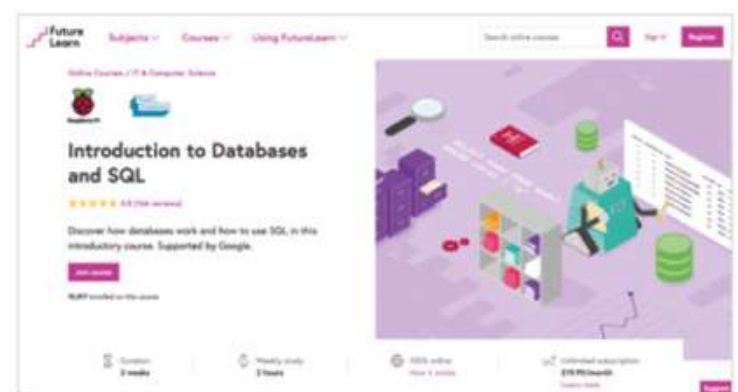
Unlike simpler ways of storing data such as spreadsheets, databases have the advantages of easy access, data integrity, and security. They also form a key part of computer science, since they underpin everything from search engines to social media.

Created by the Raspberry Pi Foundation, and aimed at complete beginners, this three-week online course starts by introducing the concept of databases and how to start

using them. It then moves on to using SQL (Structured Query

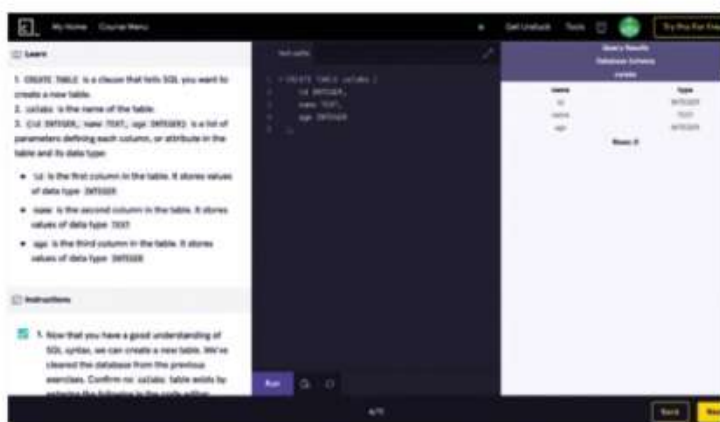
Language) to retrieve, filter, and modify the data in a database (making use of primary keys). The final week covers how to modify an existing database, along with exploring the relationships between tables and how they can be joined. As well as video and text tutorials,

and quizzes, there are practical activities with example SQLite databases using either the **sqliteonline.com** website or the DB Browser application. It all adds up to an excellent introduction to databasing and SQL. 



Online courses

Hone your databasing skills with these web courses



LEARN SQL

This interactive Codecademy course covers the SQL basics in four sections: Manipulation, Queries, Aggregate Functions, and Multiple Tables.

► magpi.cc/codecademysql

INTRODUCTION TO DATABASES AND SQL QUERYING

Udemy's short step-by-step course for complete beginners will help

you to become acquainted with SQL syntax and querying.

► magpi.cc/udemysql

SQL FOR DATA SCIENCE

Available on a seven-day free trial, this primer in the fundamentals of SQL will enable you to start analysing data for data science purposes.

► magpi.cc/courserasql

Learning SQL

AUTHOR


Alan Beaulieu

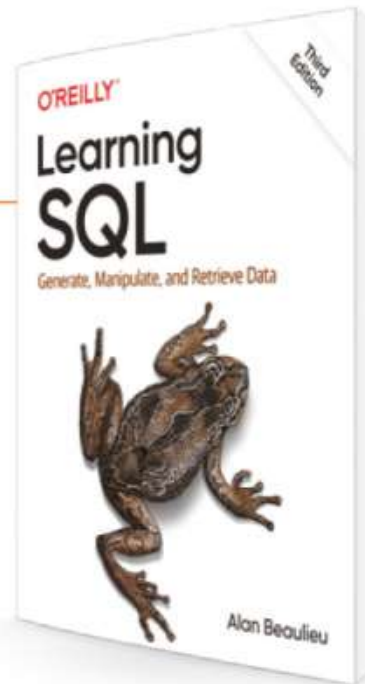
Price:
£48 / \$60

magpi.cc/learningsql

The SQL programming language is a widely used tool for interacting with data. Now on its third edition, this 380-page guidebook helps you to unleash the power of SQL. Suitable for beginners, it covers all the essential features, starting off with the basics of creating a database and using SQL data statements to generate, manipulate, and retrieve data. You then learn how to create database objects such as tables, indexes, and constraints with SQL schema statements, followed by a primer on using queries. Later topics include

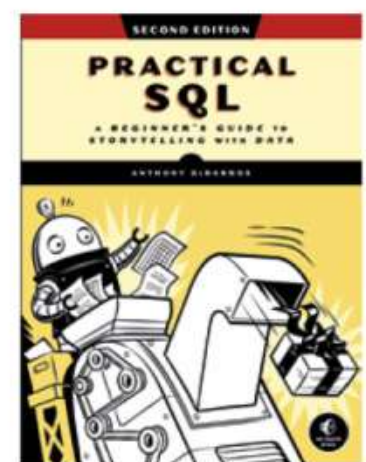
filtering using conditional logic, learning about joins, and using aggregate functions.

MySQL (part of the LAMP stack for web applications) is used throughout for interactive examples, but it tells you if a command is different for other SQL-based database software. 



Essential reading

Useful books for honing your database skills



PRACTICAL SQL

Now on its second edition, this 392-page guide offers a hands-on approach with lots of real-world examples and applications as you 'find the story within data'.

► magpi.cc/practicalsql

DATABASE DESIGN FOR MERE MORTALS

Ideal for novices learning database design, and also a useful refresher for others, this guidebook is easy to follow and covers all the fundamentals in a software-independent fashion.

► magpi.cc/ddmeremortals

SQL ANTIPATTERNS

A useful reference guide for more experienced SQL database designers, it addresses a host of 'antipatterns' – erroneous practices that result in poor results – and how to fix them.

► magpi.cc/sqlantipatterns

LibreOffice Base Documentation

AUTHOR


The Document Foundation

Price:
Free

magpi.cc/lobasedoc

There are numerous database packages available for Raspberry Pi, including the one built into the LibreOffice productivity suite (installable via Recommended Software). Called 'Base', it's a relational database application

based on the HSQL Database Engine, but can work with files in most database formats such as MySQL/MariaDB and Microsoft Access.

The official documentation gives you a thorough grounding in Base, starting with useful advice for how to plan a new database before creating it. Following that, it covers creating tables (via Design View or the wizard) and defining relationships between them. It then moves on to creating and modifying forms for data entry, followed by creating search queries and reports. Finally, it covers how data sources can be used with other LibreOffice components. 





Alex – Super Make Something

Robotics, engineering, DIY projects – Alex uses Raspberry Pi a lot

- Name **Alex Schepelmann** | ► Occupation **Engineer**
- Community role **YouTuber** | ► URL **magpi.cc/supermakesomething**

Back in issue 91 (magpi.cc/91), we covered an amazing LED mirror from Alex Schepelmann, aka Super Make Something. That was about two and a half years ago, and the YouTuber has been making even more cool stuff in that time.

“Super Make Something is a YouTube channel that aims to teach people about 3D printing, electronics, robotics, and other STEM topics through step-by-step DIY project videos,”

Alex explains. “The source code and design files for all of my projects are always released for free alongside my videos, and while it is always cool to see others reproduce my projects, my real hope is that viewers can use bits of code or parts of my designs as a starting point for their own builds.”

What is your history with making?

I have always been interested in making since I was little. One of

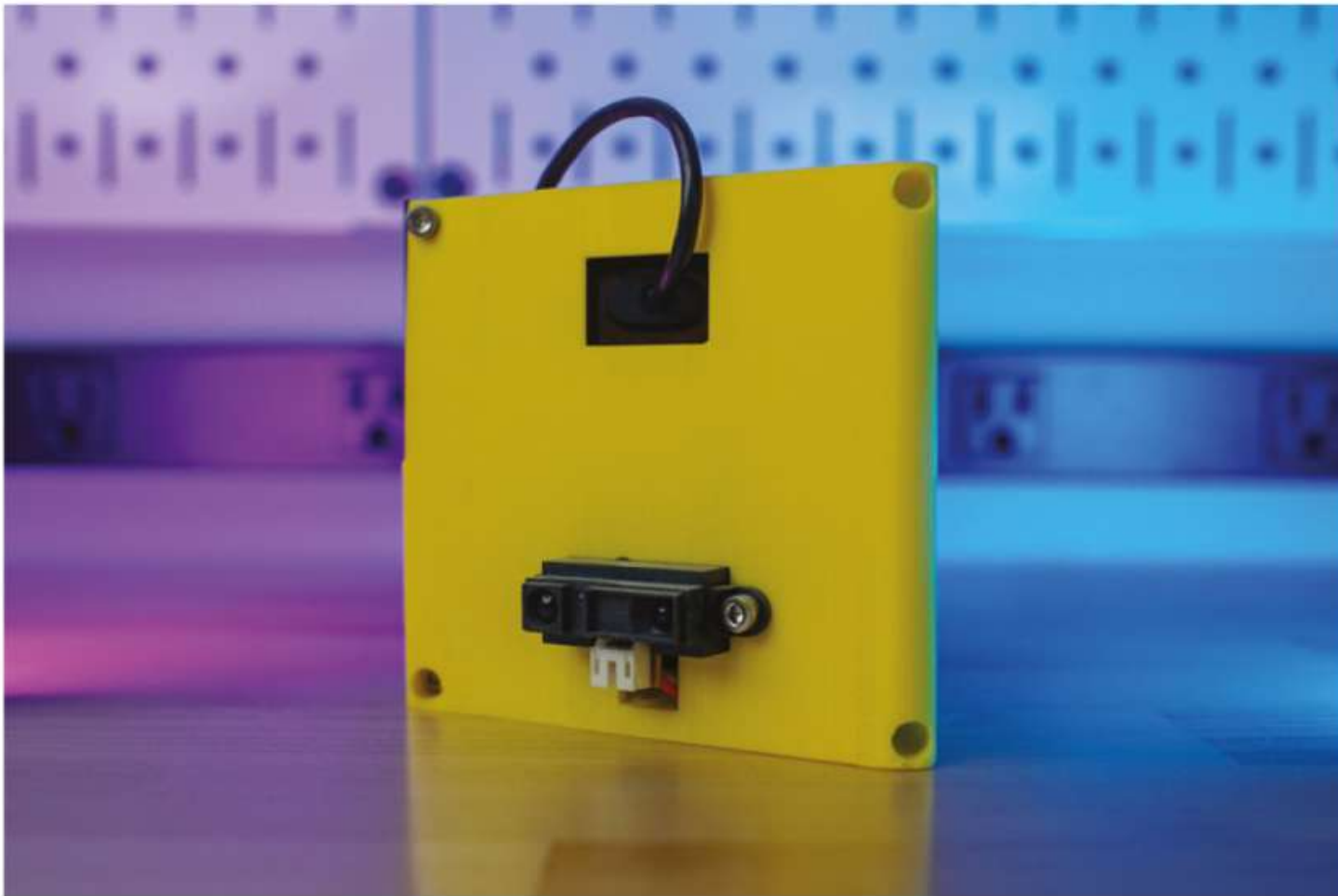
my favourite hobbies growing up was to draw and craft. When I was five, I took apart my grandmother’s wristwatch to see what was inside, and was fascinated by all of the gears and intricate moving parts. (Unfortunately I wasn’t quite able to get it back together, and the watch ended up running fast.) This interest in mechanisms, as well as a healthy appetite for science-fiction books, movies, and video games eventually led me to study Mechanical Engineering as an undergraduate and Robotics in graduate school.

When did you learn about Raspberry Pi?

I first learned about Raspberry Pi when it launched in 2012. During graduate school, I was enrolled in a Systems Engineering class in the 2012 spring semester and one of the early class assignments was learning how to program a microcontroller. I had only programmed on computers before, and thought that embedded electronics were absolutely fascinating – there were so many things that tiny computers could be useful for, since you could literally put them anywhere! After working my way through the




► The LED mirror was a big hit at Maker Faire



◀ LunchBot 9000 has helped Alex save loads of money on lunches at work

assignment, I wanted to make a project that involved video output, but quickly learned about the computational limitations that microcontrollers can have. About a week later, my office mate excitedly told me that a new \$35 computer was releasing, which absolutely blew my mind. That computer was the original Raspberry Pi!

a significant difference between a template image and whatever the camera was currently seeing, my code would light up the LEDs. This would then let me know if people were sneaking up behind me at my desk – I had a few jump scares when working in the grad school lab late at night and didn't think that anyone else was around, haha.

and its WiFi functionality, along with an IR distance sensor and a 3D-printed case, this device plugs into an outlet near my front door and sends a tweet that pops up on my smartwatch to remind me to take my lunch when it senses motion between 6:00 and 8:00 am. That project has definitely paid for itself multiple times over by saving me trips to the cafeteria at work. 

“ My office mate excitedly told me that a new \$35 computer was releasing, which absolutely blew my mind ”

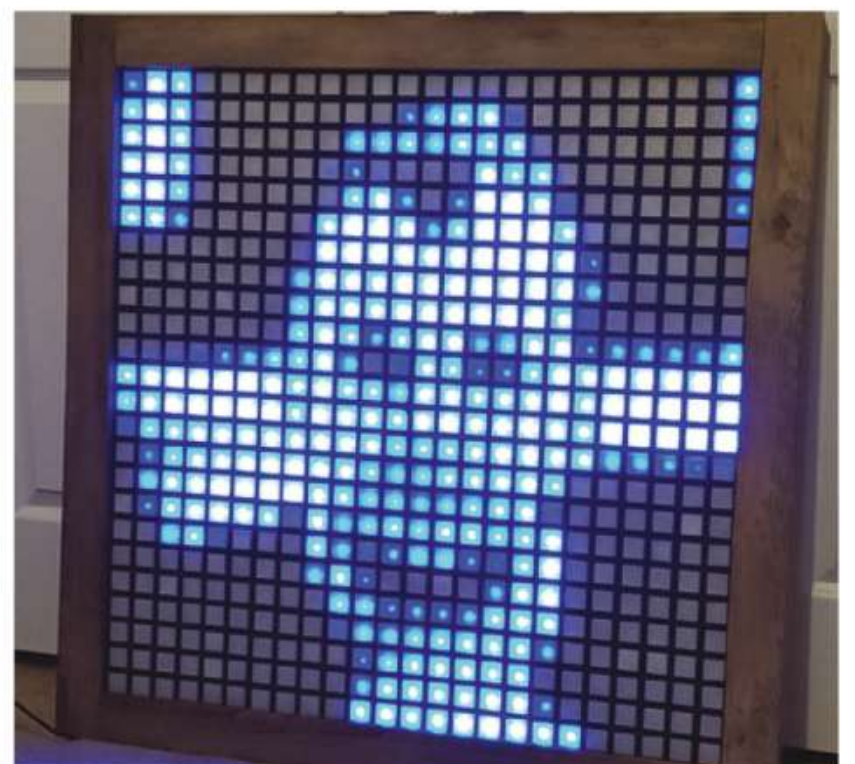
What was your first Raspberry Pi project?

After receiving my first Raspberry Pi, I spent a lot of time learning Linux and Python, and also explored many projects that were being developed for the system, like RetroPie. My first independent project was to use the Raspberry Pi Camera and some NeoPixel LEDs to create a rudimentary motion detector based on background subtraction. Whenever there was

What's your favourite Raspberry Pi project that you've made?

It's a tie between the Giant NeoPixel LED mirror that I built for the 2019 Cleveland Maker Faire and 'LunchBot 9000', a tweet-sending motion detector that uses a Raspberry Pi Zero.

'LunchBot 9000' is a more practical 'everyday' build that came about because I kept forgetting to take my lunch to work. Using a Raspberry Pi Zero



▲ You can follow the build online and make one yourself

Electromagnetic Field 2022

After four years, the UK's premier festival for makers, creatives, and the curious returns.

PJ Evans packed his tent and went looking for a certain small computer



► Brian's digital zoetrope wowed visitors, who could put their own images on the e-ink screens
Photo credit: Michael Horne

Following cancellation in 2020, **Electromagnetic Field** has made a triumphant return to its home in Eastnor, near the Welsh border. Early June saw nearly 3000 attendees come together to enjoy talks, workshops, socials, and a nightlife of music, films, laser shows, and a mysterious abandoned underwater base known as 'Null Sector'.

Over three days, a packed programme saw a wide breadth of presentation topics, from the history of railway safety to restoring electron-scanning microscopes. Meanwhile, the 'villages' (groups of people with a common interest camping together) displayed hundreds of projects. Naturally, Raspberry Pi models of all types were in attendance.

Visualising radio noise and playing games

An installation by Daniel Jones, 'Aerials', monitored the surrounding space for WiFi, Bluetooth, and other radio signals. These were processed by a Raspberry Pi 4 and converted into light and sound on three metal towers. As visitors



- ▲ Nearby radio signals were captured and turned into light and sound by a Raspberry Pi 4
- ◀ By day, talks and workshops. By night, Null Sector awakens and the skies fill with lasers

approached them, they could see the patterns change as their phones were detected.

In Null Sector, the cyberpunk aesthetic was driven by several Raspberry Pi computers. This included an RFID-based game where players had to find different check-in points to capture those devices for their team. The game was created by Michael Turner, who used a Raspberry Pi 4 to control the game, using MQTT to talk each device. A further Raspberry Pi controlled game registration.

“ Topics from the history of railway safety to restoring electron-scanning microscopes ”

I'm spinning around

In the lounge area, Raspberry Pi frequent-flyer Brian Corteil unveiled his latest project, Zoe. This is a fully functioning zoetrope, one of the earliest forms of animation, reimagined using

Pimoroni Badger e-ink screens. Not only did we love the sheer scale of the build and the beautiful engineering, but we also marvelled at the time and care taken to allow people to place their own images on it, using just an A4 piece of paper and a scanner. Kids delighted in creating their own animations.

If you're interested in attending in future, follow [@emfcamp](#) on Twitter. Electromagnetic Field will be back in 2024, and we can't wait to see what the Raspberry Pi community create next! 




- ◀ Electromagnetic Field bills itself as "A weekend of geekery in an internet-connected field" and will return in summer 2024

MagPi Monday

Amazing projects direct from our Twitter!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

Here's a selection of some of the awesome things we got sent this month – and remember to follow along at the hashtag #MagPiMonday! 

01. Another week, another cool robot from Kevin
02. Have you ever seen a robot with a top hat? Love this Omnibot build
03. This shifty-eyed alarm clock will help you wake up
04. This is a very cool start to a cool-sounding camera project
05. A great bit of upcycling! We'd probably have to keep it to ourselves too
06. We're not sure what's better, the classy case or the jazz
07. More amazing synth work from diyelectromusic
08. A very merry unbirthday to your nephew
09. We're really happy to see this project finally complete! It looks amazing





Kevin Just
@juztKevin87

03

Replying to @TheMagPi

Sure thing 😊

I added a wake up animation to my pico clock ⚙️

(tweet containing a video of the full animation
[twitter.com/juztKevin87/st...](https://twitter.com/juztKevin87/status/1548888888))



Alex Hollings
@AlexHollings52

05

Replying to @TheMagPi and @Raspberry_Pi

Turned a tissue box into a 90's Nickelodeon streamer I pretended was for my daughter but now actually lives in my office 😂



Tom Burwood
@TomBurwood83

08

Replying to @TheMagPi

My Mad Tophat was a hit at my nephews wedding on Saturday!!



JamHamster
@RealJamHamster

09

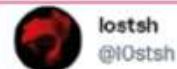
Replying to @TheMagPi

I've finally, after 4 months, finished my latest Pi build. It was a bit of a mission and used every skill I had (and some I didn't and had to learn!) 😊

JamHamster @RealJamHamster · May 22

I've finished my curved screen CRT emulator! 🎮
It has an upscaler to play original systems and an internal Pi 4 with an SD card slot in the back to change operating systems.
It works way better than expected and looks rather nice too! 🥳

[Show this thread](#)

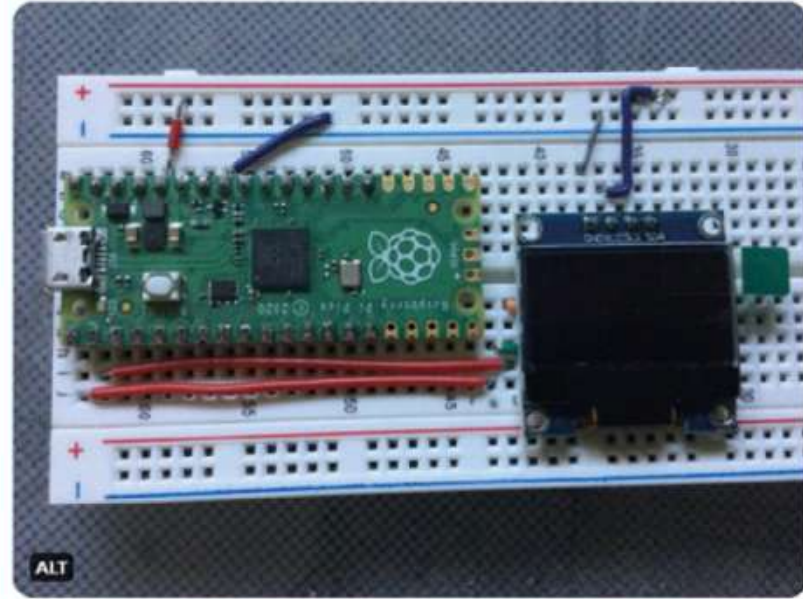


lostsh
@l0stsh

04

Replying to @TheMagPi

LCD smart driver to build a PI based camera



diyelectromusic
@diyelectromusic

07

Replying to @TheMagPi

I used my Pico as a simple vintage synth "patch loader" over MIDI.

I'm still waiting for someone to sell me a simple Pico serial MIDI interface btw so I don't have to keep using my diy one... #hint :)

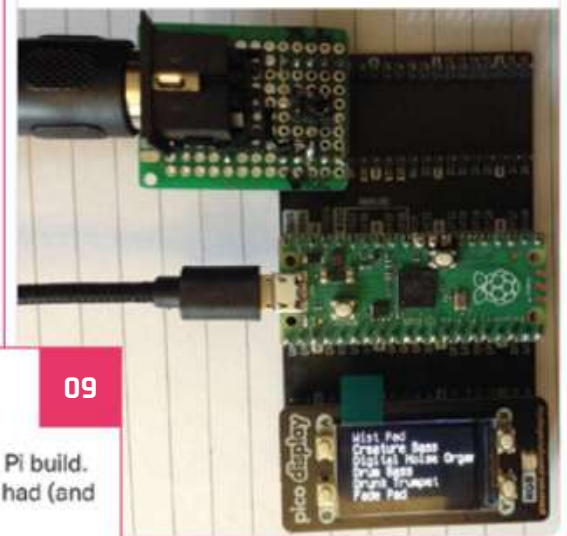
#MagPiMonday

diyelectromusic @diyelectromusic · May 28

Using a @Raspberry_Pi #PiPico, #CircuitPython and a @Pimoroni Pico Display as a simple #MIDI SysEx patch "loader" for a vintage synth - in this case a Casio CZ 3000.

[diyelectromusic.wordpress.com/2022/05/28/ras...](https://diyelectromusic.wordpress.com/2022/05/28/ras-...)

[Show this thread](#)



Jon Herd
@jonherd

06

Replying to @TheMagPi

Finally got this internet radio working with Hifiberry DAC and Adafruit 3.5 TFT touchscreen!





Lewis Workshop
@LewisWorkshop

10

Replying to @TheMagPi

I've been trying to decide whether to resurrect this pi based 3d scanner, or just break it for parts. #MagPiMonday



Dr Footleg - Roboteer
@drfootleg

...

Replying to @TheMagPi

This weekend I finished the assembly and coding of my school teams #PiWars robot. #MagPiMonday



12

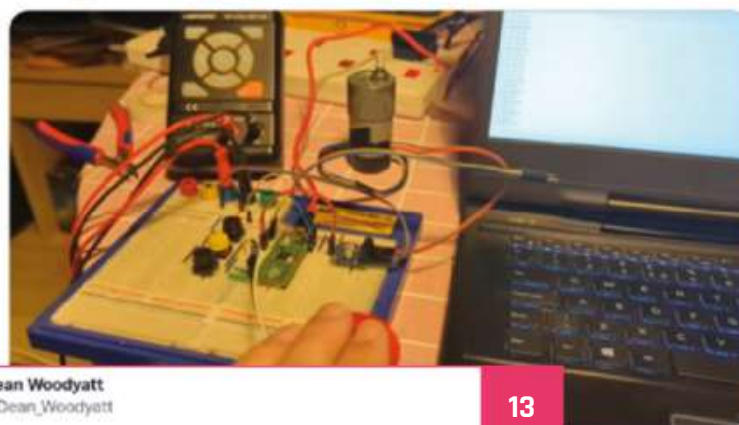


Brian Cortell
@CannonFodder

11

Replying to @TheMagPi and @GetPiTop

Still working on my Zoetrope project, been working on the controls for spinning Zoe. Wiring loom, buttons in place. Breadboarded the controls and motor controller for testing and programming the Pico. #ZoetropeMadScience #MagPiMonday



Dean Woodyatt
@Dean_Woodyatt

13

Replying to @TheMagPi

Pi pico based Amiga A500Mini working keyboard! This first prototype proved the concept, second prototype, incorporating changes to be a proper sellable product is under development. Schematics and code will be public, keycap CAD and gerbers won't be for a year or two.

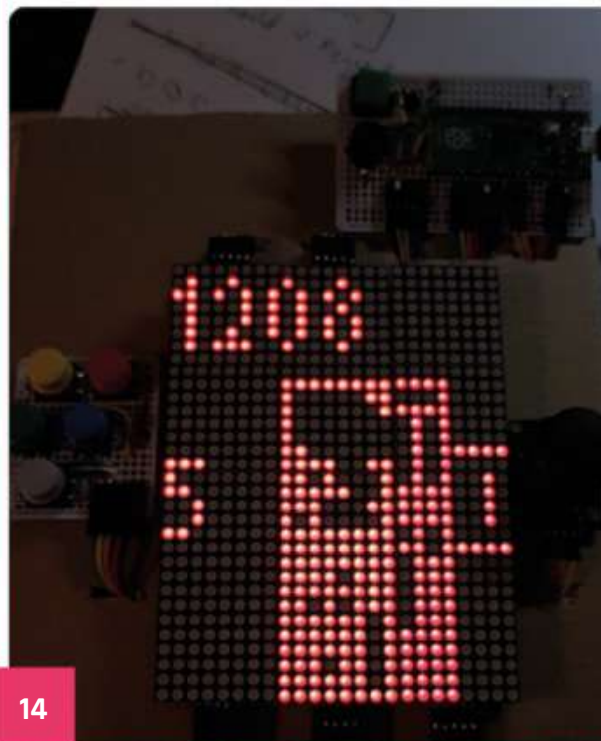


Pierre Pollakoff
@ONLUPF

...

Replying to @TheMagPi

Still working on my #pipico based game console: #picovision . Tetris game is now finished. I will now develop the breakout game



14

10. We say keep scanning! There aren't many of them around
11. You can see how well Brian's new zoetrope works in PJ's EMF report
12. We would love to see this digging robot in action
13. This tiny Amiga is very cute
14. We like the aesthetics of this Tetris game – a breakout clone will also be great!

Crowdfund **this!**

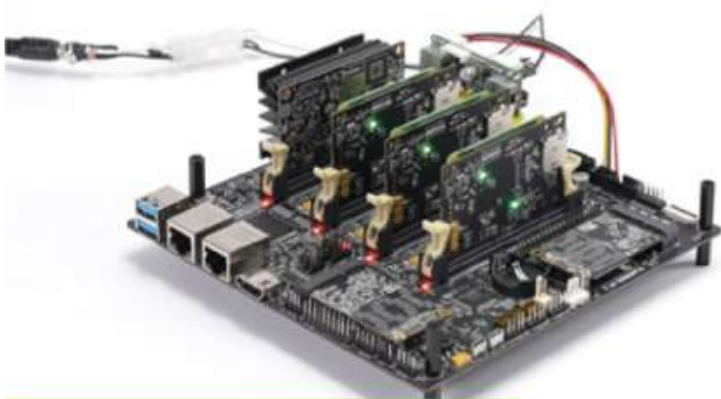
Raspberry Pi projects you can crowdfund this month



USB cases for Raspberry Pi Zero 2

These cases can be used to simply and quickly plug in a Raspberry Pi Zero 2 W, and one of them allows you to plug in an original PlayStation or PlayStation 2 controller so you can use it with RetroPie!

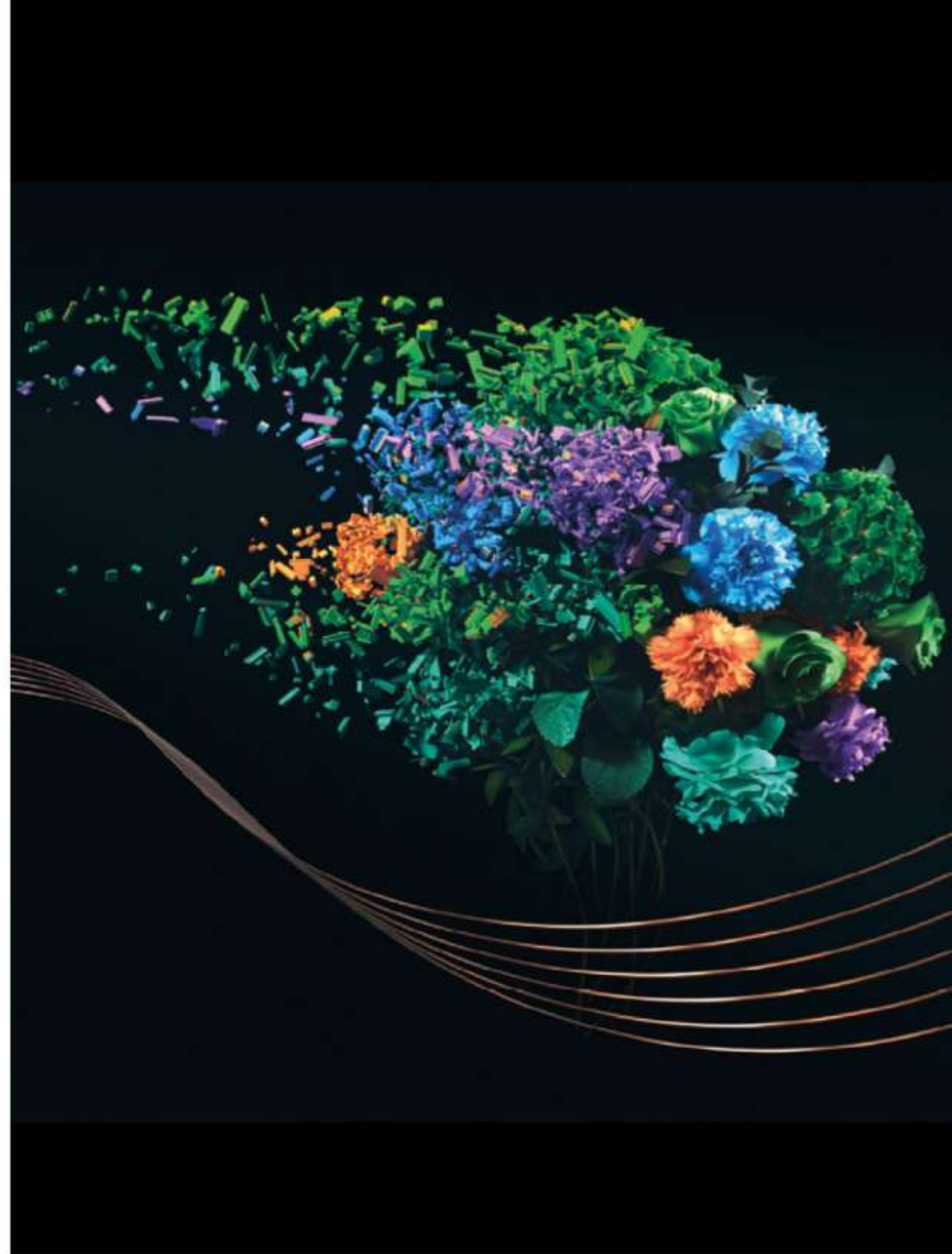
► kck.st/3sNWUDx



Turing Pi 2 Cluster

"The Turing Pi 2 is a four-node mini ITX cluster board with a built-in Ethernet switch that runs Raspberry Pi CM4 or Nvidia Jetson compute modules in any combination. It's like a data centre server rack but compact, noiseless, and highly energy-efficient. It's perfect for building your own Homelab, self-hosting, running the cloud-native stack (e.g. Kubernetes), and machine learning 24/7. The use cases are limited only by your needs and imagination."

► kck.st/3yFWMtx



50 Years of COMBICON

The Spirit of Connecting

50 Years of variety!

We are proud of the past, and we welcome the future. COMBICON, the world's largest portfolio of PCB connection technology, is celebrating its 50th anniversary. A magnificent success story – written alongside you as our customer and partner. Celebrate this exceptional birthday with us.



For additional information call 01952 681700 or visit phoenixcontact.com/combicon50years

Your Letters



Argon stock

I am interested in building your Raspberry Pi based 'Ultimate' server. I have seen some press releases regarding the Argon EON Enclosure which I believe could provide the external storage for this project, I already have the Argon40 Case with an M2 SSD fitted. However, I have had no success in finding a UK-based supplier of the Argon EON enclosure, only a USA-based one who wants in excess of £34 for postage. Grateful if you could advise me where to purchase one in UK.

Barry via email

At the time of writing, it looks like they're currently out of stock in the UK. However, you can sign up for notifications from The Pi Hut to see when they're back in stock. They would definitely be a good hard drive enclosure when available, though.



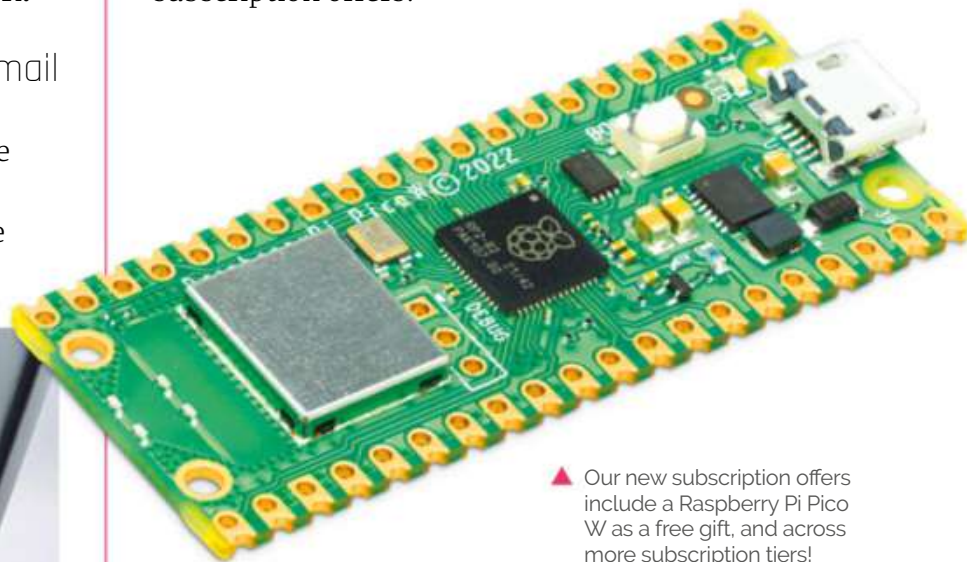
▲ Argon EON cases can hold several hard drives at once for an excellent NAS

Subscription gifts

What subscription do I need to get to receive the free gift? Is it different per length of subscription? I'd like to know for a gift.

Ploy via Twitter

We've just updated our subscription offer to include a Raspberry Pi Pico W, which you can read more about on page 28. As well as getting a gift with a twelve-month print subscription, you can also now get a Pico W with our six-month and 3 for £10 subscription offers!



▲ Our new subscription offers include a Raspberry Pi Pico W as a free gift, and across more subscription tiers!

Contact us!

- Twitter @TheMagPi
- Facebook magpi.cc/facebook
- Email magpi@raspberrypi.com
- Online forums.raspberrypi.com

3 ISSUES FOR £10



▲ The recent Metrocentre pop-up was a great time

Pop-up again

I missed the pop up store in Gateshead – well, I live in Brighton so it was a bit far for me. Are there any more pop-ups coming in the future? Or should I just come to Cambridge for a nice weekend?

Alex via email

There are two more pop-ups planned right now – the next one is in July in Edinburgh, which is a bit farther away. However, there is a London pop-up in October. Here are the details:

St James Quarter, Edinburgh
Friday 29th & Saturday 30th of July

Oxford Street, London
Friday 28th & Saturday 29th of October

The Cambridge store is permanent though, so you can always visit it!



FREE
RASPBERRY PI
PICO W

☎ Subscribe by phone:
01293 312193

✉ Subscribe online:
magpi.cc/subscribe

✉ Email: **magpi@subscriptionhelpline.co.uk**

Subscribe for £10 is a UK-only offer. The subscription will renew at £30 every six months unless cancelled. Free Pico W included with a 12-month subscription in USA, Europe and Rest of World. This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.

Wireframe

Join us as we lift the lid
on video games

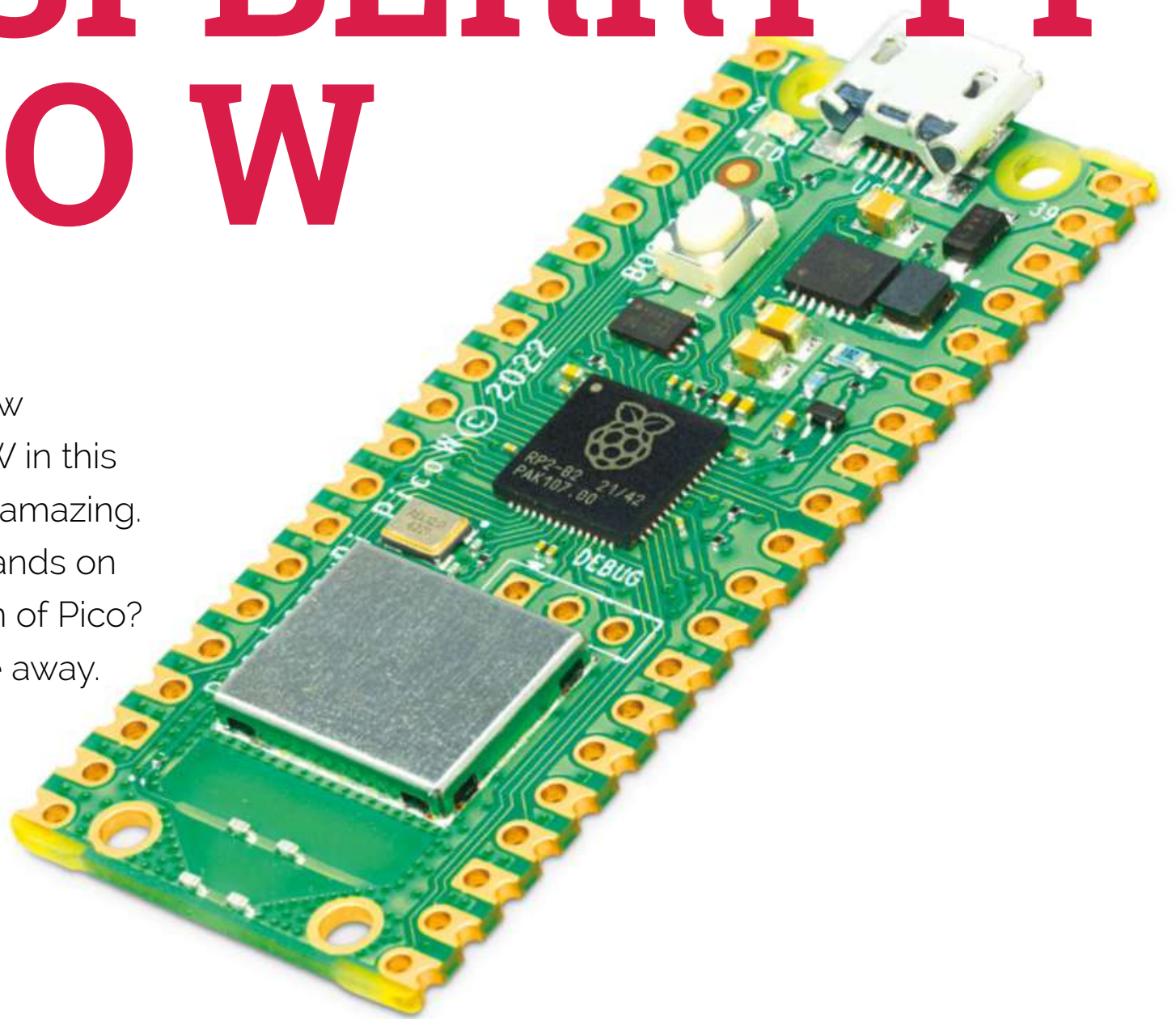


Visit wfmag.cc to learn more

WIN 1 OF 20

RASPBERRY PI PICO W

You've seen the new Raspberry Pi Pico W in this issue and it is fairly amazing. Want to get your hands on the wireless version of Pico? We have 20 to give away.



Head here to enter: magpi.cc/win | Learn more: magpi.cc/picow

Terms & Conditions

Competition opens on **30 June 2022** and closes on **28 July 2022**. Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

HackSpace

TECHNOLOGY IN YOUR HANDS

THE **MAGAZINE**
FOR THE **MODERN MAKER**



SUBSCRIBE AND
SAVE UP TO
35%
on the cover price



ISSUE #56

OUT NOW

hsmag.cc





THE MAGPI #120
ON SALE 28 JULY

Plus!

Remarkable
Robots

CrowPi L laptop

64-bit
Minecraft Server

DON'T MISS OUT!

magpi.cc/subscribe

TWITTER @TheMagPi

FACEBOOK fb.com/MagPiMagazine

EMAIL magpi@raspberrypi.com

EDITORIAL

Editor

Lucy Hattersley
lucy@raspberrypi.com

Features Editor

Rob Zwetsloot
rob@raspberrypi.com

Sub Editor

Nicola King

ADVERTISING

Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

DESIGN

criticalmedia.co.uk

Head of Design

Lee Allen

Designers

Olivia Mitchell, Ty Logan,
Sam Ribbits

Illustrator

Sam Alder

CONTRIBUTORS

David Crookes, PJ Evans, Ben
Everard, Rosemary Hattersley,
Nicola King, Phil King, Sean
McManus, Stephen Smith,
Mark Vanstone

PUBLISHING

Publishing Director

Russell Barnes
russell@raspberrypi.com

Director of Communications

Liz Upton

CEO

Eben Upton

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
magpi.cc/subscribe
magpi@subscriptionhelpline.co.uk



This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

The MagPi magazine is published by Raspberry Pi Ltd, Maurice Wilkes Building, St John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under

a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).
ISSN: 2051-9982.





Taking it to the wireless

Pico W offers the chance for a whole new set of maker experiences. By **Lucy Hattersley**

When I first got my Raspberry Pi I was hugely excited by the prospect of things you could make with technology. To this day, it's the possibilities offered by Raspberry Pi that interest me, in some ways more than the reality.

A lot of makers, including our very own PJ Evans and HackSpace's Ben Everard, recently attended EMF (Electromagnetic Field, emfcamp.org). They came back energised and enthusiastic and packed with ideas for things to make. You can read more about this excellent event on page 86.

Over the years, I've built earthquake detectors, laptops, tablets, wheeled robots, walking robots, smart speakers, countless consoles, and all manner of smart home and garden devices.

I think my favourite robot remains the simple wheeled buddy I built with an old tin box and my first-generation Raspberry Pi. So much so that I've asked PJ to recreate a similar tutorial for our robotics special. Apologies, but this will now be in issue 120 after Raspberry Pi Pico W blasted into take-off.

Pico W is the most exciting new Raspberry Pi product I've tested in a long time. In many ways, the small RP2040 development platform

exceeds its bigger Raspberry Pi computers. Pico is great fun for coding and making projects is faster and, often, easier. It's simple to set up and you don't need as many components, such as a microSD card and operating system.

I've recently received a Raspberry Pi Pico wheeled robot kit by Kitronik (magpi.cc/kittronikrobot) It is going to be epic to rediscover robotics from a Pico W perspective: wheeled robots

“ My favourite robot remains the simple wheeled buddy I built with a tin box ”

make good use of wireless networks. I'm hoping to implement a web-based interface to control it.

I've also received a CrowPi L laptop kit, which is the latest effort to turn Raspberry Pi 4 into a laptop and learning system. It's thinner and lighter than the CrowPi 2 and has a different approach to integrating electronic components and Raspberry Pi's GPIO pins.

Laptop overload

I own a fairly wide selection of computers, from classic Macs and

ageing Lenovo ThinkPads to up-to-date Intel- and ARM-based laptops and tablets. They're all fun in a way.

None of those other computers is remotely as interesting as Raspberry Pi. They do things, for sure. But they tend to be things somebody else designed them to do. Raspberry Pi, on the other hand, enables me to be the designer, developer, and maker. OK, some of my designs work better than others, but each one is a learning experience.

I can't wait to learn from making things with Pico W. We've got a new electric bus in the neighbourhood, and I'm looking forward to building a bus tracker that sucks up data from the API and lets me know when it's about to arrive. I also want to install a local temperature sensor to keep a closer eye on the heat and humidity levels in our kitchen and bounce back the data to my Raspberry Pi 400.

Whatever happens, I'm sure I will be spending the new months making incredible gadgets with Pico W. I, for one, can't wait. I can't wait to see what you all make with it as well. ”

AUTHOR Lucy Hattersley

Lucy is editor of *The MagPi* and has named each and every robot. Suggestions for the wheeled Pico W are welcome.

magpi.cc

HIGHPI PRO

———— The new case from the HiPi.io team ————



- Rapid tool-free assembly and disassembly
- Large internal volume for HATs
- Compatible with Pi 2/3/4
- Multiple lid options
- Passive & Active Cooling options
- Secure microSD card cover
- VESA mount support
- Molding-configurable output ports customizable for volume orders
- Printed logo for your branding

Available at these great Pi stores:

PiKVM

Manage your servers or workstations remotely

A **cost-effective** solution for data-centers, IT departments or remote machines!



PiKVM HAT
for DIY and custom projects



Pre-Assembled version

- Real-time clock with rechargeable super capacitor
- OLED Display
- Bootable virtual CD-ROM & flash drive
- Serial console
- Open-source API & integration
- Open-source software

Available at the main Raspberry Pi resellers



Reseller suggestions and inquiries:
wholesale@hipi.io